UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

ANOTACIÓN DE VÍDEOS PARA EL ANÁLISIS DEL LENGUAJE INFANTIL

Luis Fernando Tartilán González
Tutor: Luis Fernando Lago

MAYO 2016

ANOTACIÓN DE VÍDEOS PARA EL ANÁLISIS DEL LENGUAJE INFANTIL

AUTOR: Luis Fernando Tartilán González
TUTOR: Luis Fernando Lago

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2016

Resumen

El objetivo principal de este proyecto es ofrecer una herramienta de anotación de vídeos, llamada AVUAM, a medida para el Laboratorio de Lenguaje Infantil de la Facultad de Psicología de la Universidad Autónoma de Madrid, LabLic (Véase referencia [1]) . Los requisitos finales de la aplicación se han ido estableciendo en diversas reuniones establecidas con ellos a lo largo del desarrollo de la aplicación.

Dicha herramienta servirá de ayuda a los psicólogos en el estudio de las variables implicadas en el desarrollo de la comunicación y el lenguaje para niños comprendidos entre 9 meses y 4 años de edad. Para realizar dicho estudio se realizan grabaciones (audio y vídeo) de sesiones de juego con los adultos que posteriormente se analizan teniendo en cuenta la producción verbal, los gestos, la mirada, etc. La herramienta desarrollada ayudará al análisis de los vídeos, permitiendo la sincronización y la anotación manual de los mismos utilizando un conjunto de etiquetas o categorías definibles por el usuario, y la exportación de los resultados a un formato más adecuado para su análisis posterior.

Respecto a otras aplicaciones de anotación de vídeos podemos decir que la nuestra aporta una mejor sincronización en la reproducción simultánea de vídeos, sencillez de uso, personalización por parte del usuario y lo más importante, está adaptada totalmente a las necesidades requeridas por el laboratorio de lenguaje infantil

Debido a la necesidad de poder modificar los ficheros de configuración de etiquetas el proyecto se ha dividido en dos módulos: la herramienta de anotación de vídeos y un editor XML visual que permite a los psicólogos una rápida modificación de estos ficheros.

Abstract

The main objective of this project it is to develop a video annotation tool, called AVUAM, for the Child Language Laboratory of the Faculty of Psychology of the Universidad Autónoma de Madrid, LabLic (See reference [1]). End application requirements have been established in various meetings with them throughout the application development.

This tool will assist psychologists in the study of the variables involved in the development of communication and language in children between 9 months and 4 years. To perform this study, children are recorded (audio and video) while playing with adults. These recorded sessions are then analyzed observing the verbal production, gestures, gaze, etc. The developed tool helps to analyze videos, allowing synchronization, enabling manual annotation thereof using a set of tags or categories definable by the user, and exporting the results to a format suitable for further analysis.

Regarding other applications of video annotation, we can say that ours provides better synchronization when two or more videos are reproduced simultaneously, ease of use customization by the user and, most importantly, it is fully adapted to the needs required by the Child Language Laboratory.

Because these customizable settings from an XML by the user the project has been divided into two modules: the video annotation tool and a visual XML editor allowing psychologists rapid modification of XMLs without requiring any knowledge of this language.

Palabras clave

Anotación de vídeos, psicología, comunicación, lenguaje, desarrollo, niños, sincronización, etiquetas, categorías, XML.

Keywords

Video annotation, psychology, communication, language, development, children, synchronization, labels, categories, XML.

Agradecimientos

En primer lugar, me gustaría agradecer al tutor Luis Fernando Lago por darme la oportunidad de desarrollar este proyecto y sobre todo por la paciencia que ha tenido conmigo durante el desarrollo de este, ya que ha sido una tarea que ha llevado bastante tiempo y en la que nos hemos encontrado con diversos problemas bloqueantes que eran necesarios resolver para poder avanzar y dar por finalizado el desarrollo de la aplicación.

Mostrar mi agradecimiento a los padres de Carlos, quienes nos han permitido utilizar un vídeo de su hijo grabado durante las sesiones con LabLic para realizar las diferentes pruebas realizadas durante el desarrollo del proyecto.

También me gustaría agradecer a todos los compañeros con los que he ido compartiendo vivencias a lo largo de estos años en la universidad y que sin ellos creo que hoy no estaría aquí debido a su ayuda, que me prestaban en cualquier momento y sus ánimos en los momentos que los necesitaba.

Una persona a la que también quiero mencionar es a Teresa, gracias a ella, a su paciencia, a sus ánimos y a su apoyo considero que ha sido una de las grandes partícipes de que haya llegado hasta aquí. Agradecerla también haberme soportado y aguantado los malos momentos que he podido tener.

Y, por último, y más importante, a mi familia. Sin ellos sí que no hubiese sido posible poder haber estudiado la carrera que desde pequeño deseaba estudiar. Sus incontables maneras de apoyarme, de hacerme seguir hacia delante y nunca dejarme caer a lo largo de los tropiezos que me he ido encontrando a lo largo de la carrera.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

El desarrollo del lenguaje oral es un aspecto fundamental en el desarrollo del niño/a, ya que cumple no solo una función de comunicación sino también de sociabilización. Es determinante en el desarrollo mental y en el proceso de sociabilización del ser humano. La adquisición del sistema lingüístico favorece el desarrollo del proceso mental y social, ya que pone en contacto con la realidad creando formas de atención, memoria, pensamiento, imaginación, etc. (Véase referencia [2]). Por todo esto es de gran importancia tener un gran conocimiento en esta área ya que nos permitirá conocer todos los factores y estímulos que ayudan al desarrollo del lenguaje.

El laboratorio LabLic centra sus estudios en el seguimiento de las habilidades comunicativas y lingüísticas de niños y niñas de entre 9 meses y 4 años de edad. El objetivo que tienen es realizar un seguimiento del desarrollo comunicativo y lingüístico temprano a través de distintas pruebas experimentales y tareas estandarizadas. Una parte de este trabajo consiste en analizar los vídeos grabados durante las sesiones realizadas con los niños, realizando anotaciones de los vídeos que luego son analizadas con técnicas estadísticas. (véase referencia [3]).

Finalmente, con la aplicación AVUAM se intenta facilitar este estudio, ofreciendo una herramienta de anotación manual de vídeo que mejora las ya existentes y que les permita la reproducción simultánea de uno o más vídeos y la anotación descriptiva de los diferentes eventos observados a lo largo del vídeo. Pudiendo clasificar estos eventos en diferentes categorías. Con ello se busca obtener los datos de los vídeos analizados de una manera más eficiente y sencilla para el usuario. Es una herramienta construida a partir de los requisitos establecidos en diferentes reuniones con los psicólogos por lo que está totalmente adaptada a sus necesidades.

1.2 Objetivos

El objetivo de este proyecto es desarrollar una aplicación de anotación de vídeos manual enfocada al análisis lingüístico y adaptada a los requisitos establecidos por los miembros de LabLic. De tal forma que facilite y agilice el trabajo al psicólogo durante el análisis de los vídeos.

Los objetivos importantes a destacar durante el desarrollo del proyecto son:

- La aplicación podrá reproducir hasta 10 vídeos simultáneamente. Otras herramientas existentes permiten hasta 4 o la reproducción de un único vídeo.

- Se podrán aplicar etiquetas sobre cada vídeo en un rango de tiempo.

- Posibilidad de sincronizar cada vídeo en un instante de tiempo diferente, funcionalidad que hasta ahora no implementa ninguna herramienta para la anotación de vídeos.

- Durante la reproducción y análisis de los vídeos es de gran importancia que no se produzca ningún tipo de retraso o de desincronización entre vídeos ya que

provocaría que el análisis pudiese no ser correcto, teniendo que descartar los datos obtenidos. En la aplicación utilizada actualmente por LabLic para el análisis de vídeos han detectado en ciertos momentos un retraso o desfase en os vídeos reproducidos.

- Poder exportar los datos del análisis realizado en un formato más adecuado para el análisis posterior.

1.3 Organización de la memoria

La memoria se ha organizado de la siguiente manera:

- **Parte 1:** Se realiza una introducción al proyecto en la que se describe de manera general el propósito del proyecto, la motivación y los objetivos.

- **Parte 2:** Se explican las herramientas y tecnologías utilizadas en la aplicación. También se explican las aplicaciones que ya existen y que cumplen la misma funcionalidad principal, la anotación de vídeos. Se explican las ventajas que tiene el uso de nuestra aplicación respecto a las ya existentes.

- **Parte 3:** Se detallan los requisitos funcionales, no funcionales y del sistema de ambos módulos.

- **Parte 4:** Se analiza en detalle el diseño de la aplicación, y el por qué de las diferentes elecciones tomadas durante el diseño de esta.

- **Parte 5:** Se describe como se han implementado los diferentes puntos principales de la aplicación: implementación API JMF, cómo se ha realizado la sincronización, etc.

- **Parte 6:** Se detallan las pruebas realizadas sobre la aplicación.

- **Parte 7:** En esta parte se explicarán las conclusiones a las que se ha llegado tras la finalización del proyecto y una lista de posibles funcionalidades o ampliaciones a desarrollar en la aplicación.

- **Anexos:** En el primer apartado podremos ver cómo utilizar la librería FFMPEG para convertir los vídeos al formato .mov sin que se pierda calidad de audio y de vídeo. En los dos últimos apartados encontraremos los manuales de usuario de ambos módulos: aplicación para la anotación de vídeos y editor XML.

2 Estado del arte

2.1 Análisis Lenguaje Infantil

El desarrollo del lenguaje es un aspecto fundamental en el desarrollo del niño/a, ya que cumple una función de comunicación, sociabilización con el entorno y autocontrol de la propia conducta. Es determinante en el desarrollo mental y el proceso de socialización del ser humano. Por lo tanto, es de gran importancia estudiar qué variables afectan al desarrollo de este ya que nos aportará información de cómo enfrentarnos y estimular su desarrollo.

Por ello, el propósito de nuestra aplicación es ofrecer al LabLic una herramienta sencilla de usar y con todas las funcionalidades necesarias para que puedan trabajar en el análisis de los vídeos. Con esta herramienta se desea poder obtener un análisis adecuado de los vídeos y poder extraer, finalmente, datos fiables para posteriormente ser analizados por los psicólogos.

Existen múltiples herramientas que permiten la anotación de vídeos y enfocadas en el análisis del lenguaje pero poseen deficiencias como la sincronización o retraso en la reproducción de dos o más vídeos simultáneamente, límite en la reproducción simultánea de hasta 4 vídeos o un único vídeo y la imposibilidad de poder indicar un instante de tiempo diferente en cada vídeo en la sincronización. Podemos decir que este tipo de deficiencias no se producen en nuestra aplicación. En nuestra aplicación no se produce retraso o desfase en la reproducción simultánea de dos o más vídeos, permite reproducir simultáneamente hasta 10 vídeos, valor recomendado, y nos permite indicar el instante de tiempo de inicio para cada vídeo en la sincronización. Además, ofrecemos las funcionalidades requeridas por LabLic durante el análisis de los vídeos y la exportación de los datos del análisis realizado en un formato más adecuado para el análisis posterior.

2.2 Aplicaciones para la anotación de vídeos

Actualmente existen varias herramientas gratuitas para la anotación de vídeos, tales como Open Video Annotation Project, Advene project, Ontolog, VideoAnnEx, etc. Pero hay dos que están enfocadas principalmente al análisis del lenguaje, que son ELAN y ANVIL (Véase referencia [4]). Actualmente la aplicación que está utilizando la facultad de psicología para analizar los vídeos es ELAN.

2.2.1 ELAN

ELAN es una herramienta profesional de anotación de vídeos manual realizada por el instituto MAX PLANCK de Holanda y está enfocada principalmente para el análisis del lenguaje. Está realizada con la tecnología JAVA y es compatible con los sistemas operativos Windows, Unix y Macintosh (Véanse referencias [5], [6], [7], [8], [9] y [10]).

La aplicación permite anotaciones descriptivas, donde la anotación debe ser una palabra, sentencia o cualquier descripción característica observada en el vídeo. Estas anotaciones son aplicables a un instante de tiempo del vídeo o un rango del mismo pudiendo estar contenidas en diferentes niveles o en el mismo en la línea temporal del panel de anotaciones de la aplicación. Además, podemos definir varias anotaciones sobre una misma anotación permitiendo explicar más en detalle una anotación ya realizada sobre el vídeo. En la siguiente imagen *(Figura 2-1)* podemos ver aplicadas varias etiquetas que están contenidas en dos niveles, "Gesto 1" y "Gesto 2" en nivel "Gestos" y "Mirada 1" y "Mirada 2" en el nivel "Mirada".

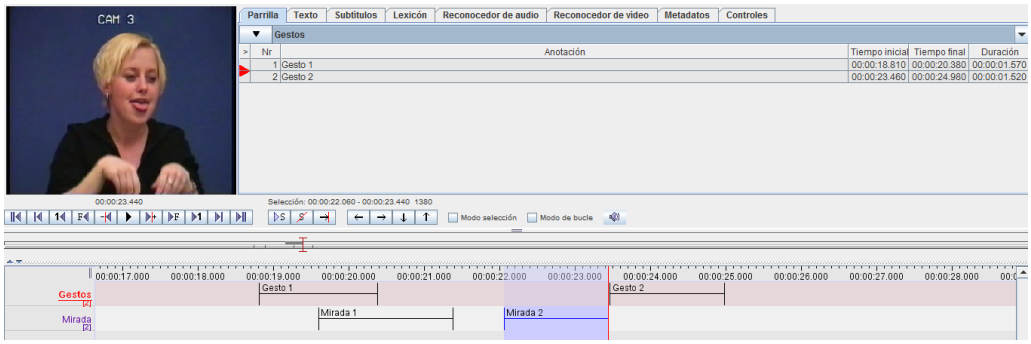


Figura 2-1: Anotación Vídeo ELAN

Es capaz de reproducir hasta 4 vídeos simultáneamente. Esta funcionalidad permite al usuario reproducir varios vídeos grabados en el mismo momento desde diferentes puntos de vista logrando una mayor visión de todo lo que ocurre. Esto proporcionará una mayor información en los análisis finales de los vídeos. En la siguiente figura *(Figura 2-2)* podemos ver la reproducción de dos vídeos simultáneamente en la que se han configurado dos niveles, un nivel por cada vídeo.

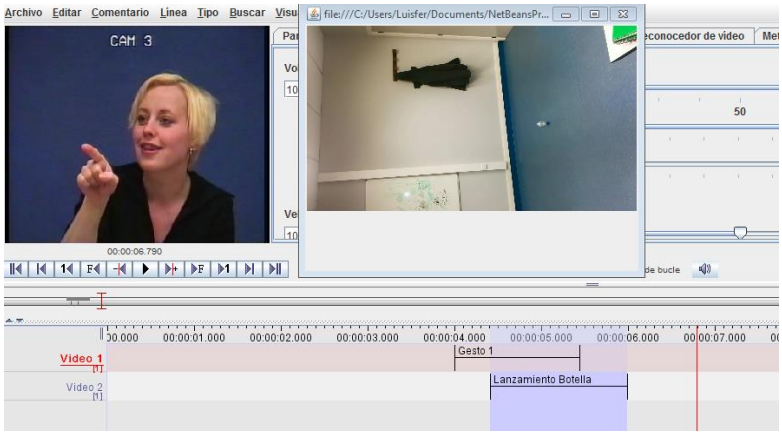


Figura 2-2: Anotación Vídeos simultáneos ELAN

La funcionalidad anteriormente descrita tiene un gran inconveniente, y es que los vídeos se sincronizan desde el momento cero del vídeo. Es decir, si justo los vídeos a reproducir en el momento inicial no están perfectamente sincronizados (audio y vídeo) se producirá un desfase durante la reproducción. Este ha sido uno de los grandes aspectos mejorados en nuestra aplicación, que permitirá la sincronización de los vídeos en cualquier punto de los mismos.

Los formatos de vídeo soportados por la aplicación son Windows Media Player, Quicktime y JMF.

Una vez terminado el análisis del vídeo la herramienta nos permite exportar la información en diferentes formatos tales como Shoebox/Toolbox, CHAT, Praat, csv/tab-delimited text files, interlinear text, html, smil y texto subtitulado.

2.2.2 ANVIL

ANVIL es una aplicación que permite la anotación de vídeos manual diseñada principalmente con fines lingüísticos. Ha sido desarrollada utilizando el lenguaje de programación JAVA por el colegio "Cognitive Science" y por el centro alemán de investigación de inteligencia artificial. Es compatible con los sistemas operativos Windows, Unix y Macintosh (Véanse referencias [11], [12] y [13]).

ANVIL soporta anotaciones descriptivas, estructurales y administrativas sobre vídeo y audio aplicables a un instante de tiempo del vídeo o un rango del mismo. Estas anotaciones las podremos realizar en la línea temporal contenida en el panel de anotación de la aplicación *(Figura 2-3)*. A diferencia, ELAN sólo soporta anotaciones descriptivas.

Esta herramienta sólo nos permite visualizar un vídeo simultáneamente lo que implica no disponer de una reproducción de un mismo escenario desde varios puntos de vista. Esta restricción es una desventaja ya que deriva en un menor nivel de detalle a poder ser observado durante el análisis de los vídeos por el usuario. Los formatos de vídeo soportados son .MOV y .AVI.

ANVIL puede importar datos de las herramientas PRATT, XWaves y en próximas versiones de ELAN. Los datos exportados gracias a esta herramienta pueden ser utilizados en aplicaciones de análisis estadístico como SPSS y Statistical.

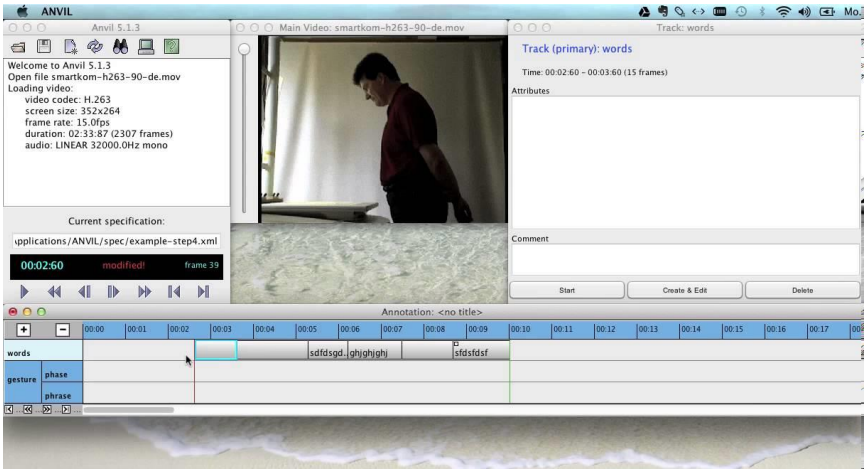


Figura 2-3: Anotación vídeo ANVIL

2.2.3 Comparativas aplicaciones

En la siguiente tabla *(Tabla 2-1)* podemos ver una comparativa de las aplicaciones de ELAN, ANVIL y AVUAM (nombre de la aplicación de este proyecto), comparando que funcionalidades implementa cada una de ellas, siendo las funcionalidades las definidas como requisitos u objetivos en este proyecto.

Funcionalidad	ELAN	ANVIL	AVUAM
Reproducción simultánea de vídeos	Hasta 4 vídeos	No soportado	Recomendado hasta 10 vídeos
Aplicar etiquetas	Soportado	Soportado	Soportado
Categorizar las etiquetas	Soportado	Soportado	Soportado
Sincronizar vídeos	Soportado	No soportado	Soportado
Sincronizar vídeos en un instante de tiempo diferente cada uno	No soportado	No soportado	Soportado
Exportar datos	Soportado	Soportado	Soportado
Customizar etiquetas	Soportado	Soportado	Soportado
Modificar y eliminar etiquetas ya insertadas	Soportado	Soportado	Soportado

Tabla 2-1: Comparativa ELAN, ANVIL y AVUAM

Como podemos apreciar en la tabla *2.2-1* nuestra aplicación aporta un conjunto de necesidades establecidas por LabLic que actualmente las aplicaciones existentes no satisfacen.

2.3 Java Media Framework (JMF)

Para el desarrollo de la aplicación se eligió el lenguaje de programación Java debido a las grandes ventajas que tiene utilizarlo: multiplataforma, gran cantidad de documentación, múltiples librerías y Frameworks libres, utilizado por una amplia comunidad de programadores y empresas, etc. Una vez elegido el lenguaje se buscó una librería multimedia realizada en Java que permitiese implementar de una manera sencilla en nuestra aplicación la reproducción de vídeos en múltiples formatos y todas las funcionalidades necesarias para controlar los vídeos. Finalmente, la elegida fue JMF debido a que cumplía todos los requisitos mencionados.

JMF es una librería realizada en Java por Sun Microsystems e IBM. Su objetivo principal es ofrecer una librería al programador que facilite el desarrollo de aplicaciones multimedia. Permite la reproducción de vídeo en formato: Quicktime (.mov) y MPEG y la reproducción de audio en formato: AU, AVI, MIDI, MPEG y WAV. Además, nos permite la grabación de audio y vídeo, streaming online mediante protocolo RTP (Real-Time Transport Protocol) y tratamiento de archivos multimedia en formatos compatibles (Véase referencia [14]).

Finalmente, durante la implementación de esta API, se pudo ver que provee un sistema de sincronización entre vídeos que permite que el control de todos ellos sea gestionado por un vídeo maestro. En nuestra aplicación no hemos utilizado esta funcionalidad ya que queremos poder sincronizar los vídeos en puntos diferentes evitando que sea necesario que los vídeos a reproducir simultáneamente empiecen en el mismo instante de inicio.

3 Análisis de requisitos

La aplicación realizada en este proyecto está destinada al uso por LabLic, por ello, ha sido necesario realizar varias reuniones a lo largo del desarrollo del proyecto para establecer los requisitos finales de la aplicación.

Tras las reuniones realizadas se decidió dividir el proyecto en dos módulos independientes:

- La aplicación principal, que es la que se encarga de la anotación de vídeos.
- Un editor XML que permitirá al usuario de una forma sencilla editar los archivos XML de configuración de etiquetas de la aplicación de anotación de vídeos.

3.1 Aplicación para la anotación de vídeos

3.1.1 Requisitos funcionales

Los requisitos funcionales establecidos en la aplicación son:

- **Reproducir uno o más vídeos simultáneamente:** La aplicación permitirá reproducir 10 vídeos simultáneamente pudiendo estar o no sincronizados.
- **Realizar operaciones básicas sobre el vídeo:** Reproducir, pausar, sincronizar, silenciar audio, avanzar o retroceder a cualquier punto de reproducción del vídeo.
- **Permitir la sincronización entre dos o más vídeos:** La sincronización entre dos o más vídeos permitirá que los controles de los vídeos esclavos sean gestionados por el vídeo maestro. Permitirá también, en el momento de sincronizar el vídeo, marcar un instante de tiempo diferente en cada vídeo sobre el que se deberá mantener la sincronización respecto al vídeo maestro. Este punto es muy importante debido a que si produce un retraso entre un vídeo y otro durante la reproducción los datos obtenidos pueden no ser fiables.
- **Poder aplicar etiquetas descriptivas sobre los diferentes vídeos que se están analizando:** Se podrán aplicar etiquetas descriptivas sobre los vídeos que se están analizando. Este punto además es totalmente personalizable por el usuario a partir de XMLs.
- **Aplicar etiquetas durante el análisis en un rango de tiempo:** Las etiquetas se podrán aplicar desde un segundo x hasta un segundo y del vídeo marcado para la etiqueta. Los tiempos tendrán que ser precisos para tener un resultado final fiable.
- **Poder agrupar las etiquetas en categorías:** Cada etiqueta estará asociada a una categoría de tal forma que el usuario pueda agrupar las etiquetas en diferentes categorías.
- **Modificar y eliminar etiquetas ya marcadas en un vídeo**: Si el usuario se equivoca o pulsa por equivocación en una etiqueta al realizar la anotación de un vídeo se le permitirá modificarla o eliminarla.

- **Exportar los datos en un formato apropiado para su posterior análisis estadístico**: Texto tabulado y texto separado por comas en formato .csv.

- **Cargar y guardar plantillas:** Permite al usuario guardar el estado actual del análisis realizado sobre los vídeos para posteriormente poder recuperarlo en cualquier momento.

- **Cargar archivos de configuración XML:** El usuario podrá cargar el archivo de configuración XML deseado desde el menú de la aplicación.

3.1.2 Requisitos no funcionales

Los requisitos no funcionales establecidos en la aplicación son:

- **Usabilidad:** La aplicación deberá ser intuitiva de manera que un usuario con pocos conocimientos informáticos sea capaz de usarla.

- **Sincronización durante la reproducción simultánea de vídeos:** La aplicación mantendrá una sincronización en la reproducción simultánea de dos o más vídeos sin que se produzca un desfase entre ellos.

3.1.3 Requisitos del sistema

Los requisitos del sistema son los siguientes:

- 200 MHz Pentium.

- Al menos 64 MB de RAM.

- Se deberá tener instalado JRE o JDK de 32 bits. Requisito para el correcto funcionamiento de la API JMF.

3.2 Editor XML

3.2.1 Requisitos funcionales

Los requisitos funcionales establecidos en la aplicación son:

- **Crear nuevos XML:** Desde esta aplicación podremos crear nuevos XML de configuración para la aplicación de anotación de vídeos.

- **Editar XML:** El usuario podrá editar XMLs de configuración ya creados.

- **Añadir y eliminar vídeos:** Dependiendo del número de vídeos que se vayan a reproducir durante el análisis el usuario deberá añadir tantos vídeos como vídeos se vayan a reproducir y/o analizar.

- **Añadir, modificar y eliminar etiquetas**: El usuario podrá añadir nuevas etiquetas, eliminarlas o modificarlas en caso de que se quiera modificar algún parámetro de alguna ya existente.

- **Poder agrupar las etiquetas en categorías:** Cada etiqueta estará asociada a una categoría de tal forma que el usuario pueda agrupar las etiquetas en diferentes categorías.

- **Guardar el XML modificado o generado:** Una vez se haya terminado de editar el XML el usuario podrá guardarlo en nuevo archivo o reemplazar uno ya existente.

3.2.2 Requisitos no funcionales

Los requisitos no funcionales establecidos en la aplicación son:

- **Usabilidad:** La interfaz de la aplicación deberá ser sencilla e intuitiva de usar. El usuario podrá utilizar las diferentes funcionalidades ofrecidas por la aplicación de una manera rápida y sencilla.

3.2.3 Requisitos del sistema

Los requisitos del sistema son los siguientes:

- 200 MHz Pentium.

- Al menos 64 MB de RAM.

4 Diseño

Debido a que la realización de este proyecto se ha dividido en dos módulos independientes, aplicación para la anotación de vídeos y editor XML, se va a detallar el diseño de cada módulo en puntos independientes.

El diseño de la aplicación para la anotación de vídeos se ha ido modificando a lo largo de su desarrollo debido a cambios en los requisitos o porque se han pedido añadir nuevas funcionalidades tras las diferentes reuniones realizas con los psicólogos de la UAM.

4.1 Aplicación para la anotación de vídeos

Es la herramienta principal y la que utilizará el usuario para la anotación de vídeos. La interfaz de la aplicación debe ser sencilla e intuitiva para el usuario, de manera que un usuario con escasos conocimientos de informática sea capaz de utilizarla.

4.1.1 Lenguaje de programación Java

Llegado el momento en el que se tuvo que decidir qué lenguaje de programación se iba a utilizar, nos decantamos por Java. Los motivos por los que se eligió este lenguaje de programación son las siguientes:

- Java es ampliamente utilizado por la comunidad de programadores actualmente y utilizado en multitud de empresas para el desarrollo de sus propias aplicaciones. Esto es debido a que es un lenguaje muy potente ya que permite realizar prácticamente cualquier tipo de aplicación independientemente del tamaño y fin de esta.

- Dispone de entornos de desarrollo (IDEs) gratuitos para el desarrollo de aplicaciones.

- Otro punto fuerte es que es multiplataforma, con esto conseguíamos realizar una aplicación que luego el usuario podría ejecutar en su ordenador sin importar el sistema operativo que utilizase. En el momento de desarrollar la aplicación no sabíamos qué sistema operativo utilizan los psicólogos para realizar la anotación de vídeos y por ello debíamos disponer de una aplicación, a poder ser, multiplataforma.

- Una de las ventajas que tiene utilizar Java es que dispone de una gran documentación, además de disponer de una gran cantidad de librerías, APIs y Frameworks libres y gratuitos que facilitan el trabajo en el desarrollo de aplicaciones.

- Finalmente, se buscaba encontrar una API para la reproducción y control de los vídeos ya que esto nos facilitaría el desarrollo del proyecto. Por ello, tras investigar en Internet se encontró la API JMF que es gratuita e implementa toda la funcionalidad requerida para nuestro proyecto en cuanto a gestión y control de vídeos se refiere.

- Nos permite ejecutar paralelamente operaciones específicas de la aplicación gracias a la implementación de varios hilos de ejecución. Esto nos será de gran ayuda en el momento de mantener la sincronización entre dos o más vídeos y que no se produzca ningún tipo de desfase o retraso entre ellos.

4.1.2 JMF

JMF es una API para el desarrollo de aplicaciones multimedia. Permite la reproducción de audio y vídeo en una gran cantidad de formatos, así como transmitir datos multimedia en stream y captura de audio y vídeo mediante micrófono o videocámara. En nuestro proyecto se utilizará para la reproducción y control de los vídeos. Esta API se eligió debido a varias razones:

- Es una de las APIs más completas y de las pocas gratuitas que se pudo encontrar en Internet.

- Está desarrollada por Sun Microsystems, ahora Oracle, por lo que daba cierta seguridad su uso y garantizaba el correcto funcionamiento de la API.

- Dispone de una gran documentación en Internet y en la propia página de Oracle.

- Es sencilla de usar e implementa toda la funcionalidad requerida por nuestra aplicación. Una de las funcionalidades de esta API es la posibilidad de sincronizar los vídeos con un vídeo maestro de manera que el vídeo maestro tome el control de los vídeos esclavos. Al ser este uno de los requisitos principales e importantes de nuestro proyecto tomo un gran peso en la decisión de utilizar esta API.

4.1.3 Modelo – Vista – Controlador (MVC)

Es un patrón de diseño de software que define la organización independiente del modelo (lógica de Negocio), la vista (interfaz con el usuario u otro sistema) y el controlador (controlador del flujo de trabajo de la aplicación).

El patrón MVC está compuesto por:

- **Modelo:**
  - Contiene la lógica de negocio de la aplicación.
  - Encapsula el estado de la aplicación.
  - Independiente del controlador y la vista.
- **Vista:**
  - Es la presentación del modelo.
  - No conocerá nada de la lógica de negocio, modelo.
- **Controlador:**

- Es el encargado de gestionar los eventos lanzados por la vista y realizar las modificaciones necesarias tanto en el modelo como en la vista.

La forma de comunicarse entre los diferentes componentes es la siguiente: La vista recibe un evento por parte del usuario, por ejemplo, pausar el vídeo, cambiar el instante de tiempo del vídeo, etc. Posteriormente el controlador gestiona este evento realizando los cambios necesarios tanto en el modelo como en la vista.

Para la realización de este proyecto se ha elegido este patrón de diseño de software debido a que al tener que desarrollar una aplicación de escritorio en la que el usuario va a estar en constante interacción con la aplicación este patrón se adapta totalmente a este proyecto. Gracias al uso de este patrón conseguimos tener el código divido en diferentes capas ofreciendo las siguientes ventajas:

- El código de la aplicación sea más legible, estructurado y sencillo de mantener en un futuro ya que cada funcionalidad de la aplicación estará separada en las tres capas comentadas anteriormente.

- Facilita la reutilización del código para otras aplicaciones.

4.1.4 Diseño clases

En el patrón a seguir durante el diseño de la aplicación se ha utilizado el patrón Modelo - Vista - Controlador (MVC). Se pueden encontrar más detalles sobre este patrón en el punto *4.1.3 Modelo – Vista – Controlador (MVC).*

En la figura 4-1 podemos ver el diagrama de clases de la aplicación AVUAM:

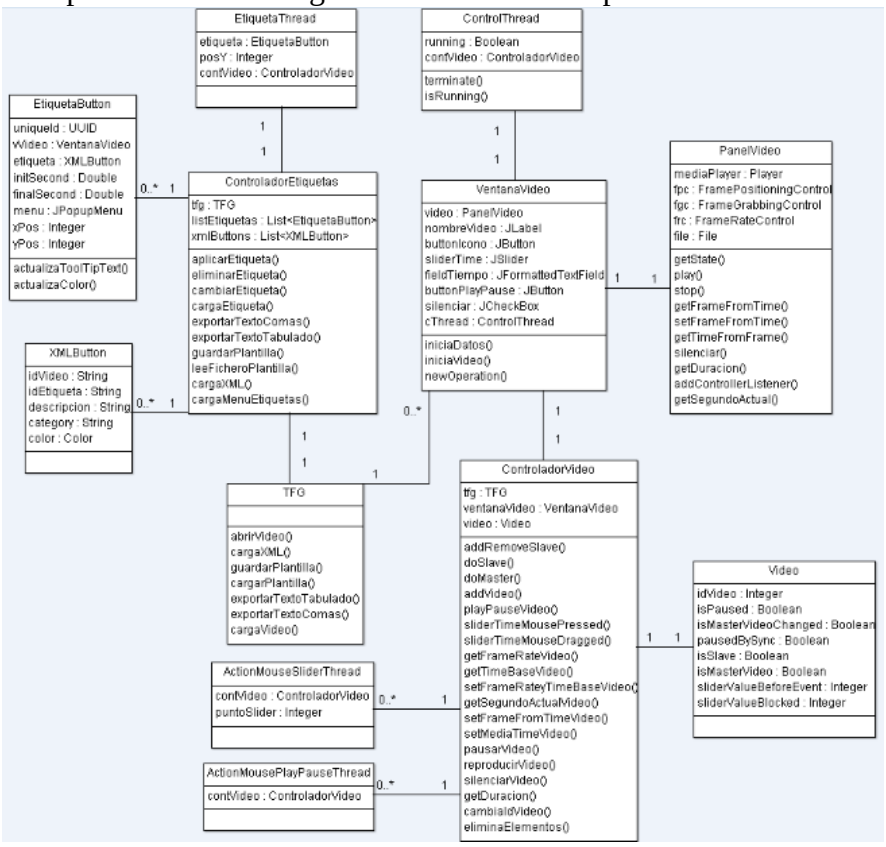


Figura 4-1: Diagrama de clases AVUAM

Siguiendo el modelo MVC las clases "TFG", "VentanaVideo" y "PanelVideo" serán las vistas. La clase "TFG" será la ventana que contenga los controles de la aplicación y los controles de los vídeos. Por el otro lado, la clase "VentanaVideo" contendrá el vídeo a reproducir que será una instancia de la clase "PanelVideo". Cada instancia de la clase "VentanaVideo" contendrá los controles propios asociados a ese vídeo, los cuales serán delegados a la clase "TFG" para ser mostrados en esta junto con los controles de los demás vídeos.

Los controladores serían las clases "ControladorEtiquetas" y "ControladorVideo". La clase "ControladorVideo" será la encargada de gestionar los eventos de los componentes de la vista "VentanaVideo", modificando los datos del modelo "Video" en caso de ser necesario. El controlador "ControladorEtiquetas" se encargará de la gestión de las etiquetas en la vista "TFG" y utilizará como modelo las clases "EtiquetaButton" y "XMLButton". También se encargará de la exportación de los datos a un fichero, carga y generación de plantillas y carga del fichero de configuración XML.

Para evitar que el controlador sea el que inicie la vista y el que detone los eventos de la interfaz ya que provoca que el código sea menos legible, el cual es uno de los objetivos del uso del patrón MVC, se ha diseñado la aplicación para que el controlador y la vista se conozcan entre ellos. De esta manera la propia vista será la que se inicie a sí misma en su método "main" y también será la que detone sus propios eventos llamando al controlador, que será el encargado de gestionar estos eventos realizando las modificaciones necesarias tanto en el modelo como en la vista.

4.1.5 Diseño de la interfaz

El diseño de la pantalla principal de la aplicación estará dividido principalmente en tres partes *(Figura 4-2)*:

- **Cabecera:** Incluirá una serie de funcionalidades en las que el usuario podrá abrir archivos de vídeo, cargar fichero de configuración de etiquetas, guardar y cargar plantillas y exportar los datos obtenidos durante el análisis de los vídeos.

- **Panel izquierdo frame principal:** Contendrá las funcionalidades para el control de los vídeos. Cada vídeo tendrá sus controles independientes. En este panel se mostrarán también las etiquetas aplicadas.

- **Panel derecho frame principal:** Es el panel de las etiquetas y contendrá las etiquetas configuradas para cada vídeo.

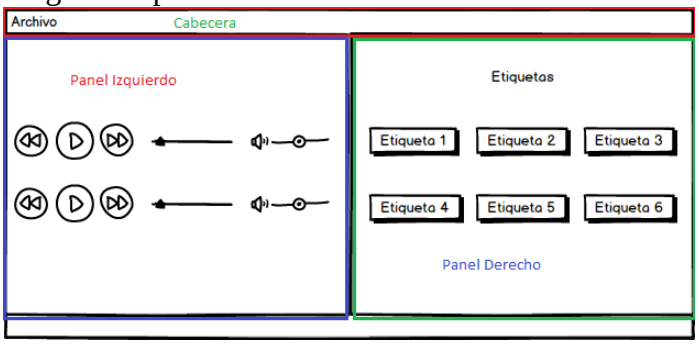


Figura 4-2: Diagrama de clases AVUAM

Cada vídeo se reproducirá en una ventana independiente. En esta ventana se mostrará el id del vídeo como título y contendrá el panel de reproducción de este *(Figura 4-3)*.

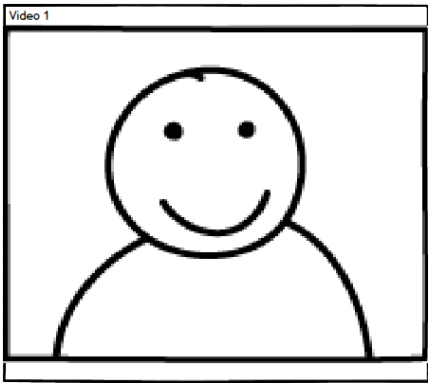


Figura 4-3: Ventana de vídeo

4.2 Editor XML

Esta aplicación permitirá editar o crear un archivo de configuración de etiquetas sin tener el usuario que cambiarlo de forma manual editando o creando directamente el fichero XML a mano. De esta forma un usuario con escasos conocimientos de informática será capaz de modificar o de crear archivos de configuración de etiquetas.

4.2.1 Diseño de clases

En el patrón a seguir durante el diseño de la aplicación se ha utilizado el mismo que en el de la aplicación para la anotación de vídeos, el patrón Modelo - Vista - Controlador (MVC). Se pueden encontrar más detalles sobre este patrón en el punto *4.1.3 Modelo – Vista – Controlador (MVC)*.

A continuación, en la figura 4-4 podemos ver el diagrama de clases de la aplicación:

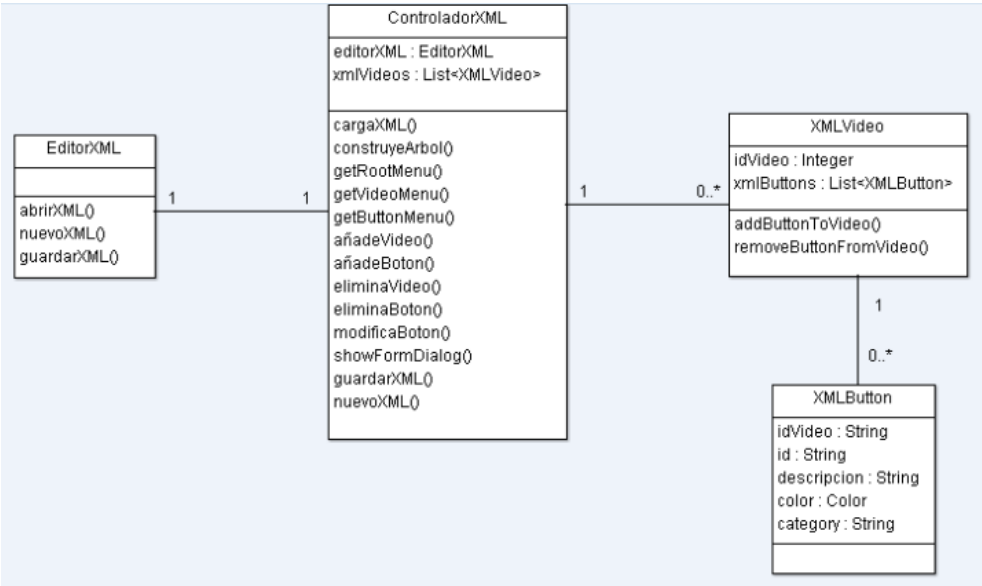


Figura 4-4: Diagrama de clases aplicación Editor XML

En este caso, siguiendo el patrón MVC, la vista es la clase "EditorXML", el modelo las clases "XMLVideo" y "XMLButton" y el controlador la clase "ControladorXML". El controlador "ControladorXML" será el encargado de gestionar los eventos producidos por la vista "EditorXML", realizando las modificaciones necesarias tanto en el modelo como en la vista.

Tal y como se ha especificado en el punto *4.1.4 Diseño de clases*, para evitar que el controlador sea el que inicie la vista y el que detone los eventos de la interfaz ya que provoca que el código sea menos legible, el cual es uno de los objetivos del uso del patrón MVC, se ha diseñado la aplicación para que el controlador y la vista se conozcan entre ellos. De esta manera la propia vista será la que se inicie a sí misma en su método main y también será la que detone sus propios eventos llamando al controlador, que será el encargado de gestionar estos eventos realizando las modificaciones necesarias tanto en el modelo como en la vista.

4.2.2 Diseño de la interfaz

El diseño de esta aplicación será una única ventana que contenga todos los nodos en disposición de árbol *(Figura 4-5)*. Tendrá un nodo raíz (nivel 1), de este colgarán los nodos de los vídeos (nivel 2) y de estos últimos colgarán las etiquetas configuradas (nivel 3).

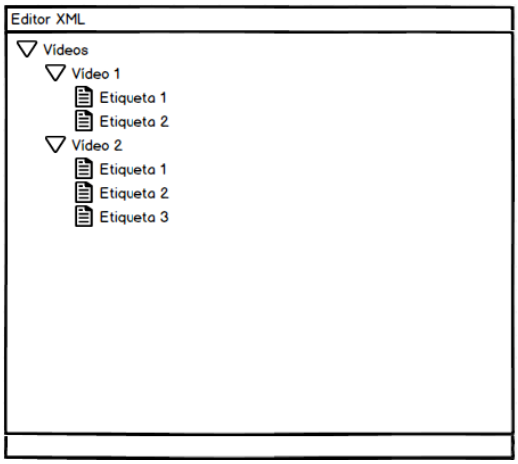


Figura 4-5: Interfaz Editor XML

Cada nodo tendrá su propio menú con el que se podrá interactuar para añadir y eliminar vídeos y etiquetas, así como la posibilidad de modificar etiquetas ya creadas. También tendrá un menú en la cabecera de la aplicación desde el que se podrá cargar y crear nuevos ficheros de configuración de etiquetas y guardar en un fichero la configuración actual de los vídeos y las etiquetas.

5 Desarrollo

Como ya se ha explicado anteriormente el proyecto se ha divido en dos módulos independientes, la aplicación principal que es la encargada de la anotación de vídeos y el editor XML que es el que permite al usuario editar los archivos de configuración XML.

5.1 Aplicación para la anotación de vídeos

La aplicación ha sido desarrollada utilizando el lenguaje de programación Java. Para la reproducción de los vídeos se ha utilizado la API JMF que nos ofrece todas las funcionalidades necesarias para la gestión y control de los vídeos en nuestra aplicación tal y como se describe en el punto *2.3 JMF*.

JMF nos provee los controles básicos sobre el vídeo, tales como pausar, reproducir, avanzar, retroceder o cambiar el instante de reproducción a través de un slider. Como la sincronización de vídeos implementada por JMF no nos proveía las funcionalidades y lógica que queríamos aplicar se decidió implementar manualmente tanto la sincronización como los controles básicos sobre los vídeos para poder tener un control total sobre los vídeos y los controles de estos.

Por cada vídeo que se abra se iniciará un hilo de control (*ControlThread*) sobre este. El hilo actualizará el campo del panel de control que indica el instante actual de reproducción del vídeo a partir del "mediaTime" de este además de comprobar si se debe de comenzar la reproducción del vídeo en caso de que se cumpla la condición de que el vídeo sea un vídeo esclavo, esté pausado, sincronizado con el vídeo maestro y se cumpla el desfase de sincronización con el vídeo maestro. Este último punto y además de los controles de pausar, reproducir, sincronizar y el slider se explicarán más detalladamente en el punto *5.1.2 Sincronización vídeos* ya que parte de su implementación viene condicionada para mantener la sincronización de los vídeos.

La aplicación, al poseer varios hilos de ejecución, en determinados momentos accede a la misma variable desde distintos hilos. Para mantener una información coherente entre los diferentes hilos y que cada hilo acceda al valor real de esa variable en ese momento se han declarado las variables que son accedidas desde diferentes hilos como "Volatile". El modificador "Volatile" indica al compilador que la variable va a ser accedida y/o modificada desde varios hilos simultáneamente y de manera asíncrona. Con esto se consigue que el valor de la variable no se almacene como una copia local en el propio hilo, sino que se obligue a acceder a la memoria principal para obtener y/o modificar el valor de la variable. El acceso al valor de las variables con el modificador "Volatile" garantiza la visibilidad del valor actualizado (Véase referencia [20]).

5.1.1 JMF

Los datos multimedia a reproducir por JMF API pueden provenir de varias fuentes:

- Archivos locales o remotos
- Vídeo o audio en tiempo real, capturando desde la propia Webcam conectada al PC.

Para poder reproducir los vídeos se debe crear un objeto Player, que es el encargado de reproducir los datos multimedia. Para crear el objeto Player debemos llamar al método

"CreatePlayer" pasándole como argumento un DataSource, un MediaLocator o la URL al archivo multimedia local.

El objeto Player se puede encontrar en varios estados durante la reproducción de un archivo multimedia. El diagrama de estados es el siguiente *(Figura 5-1)*:

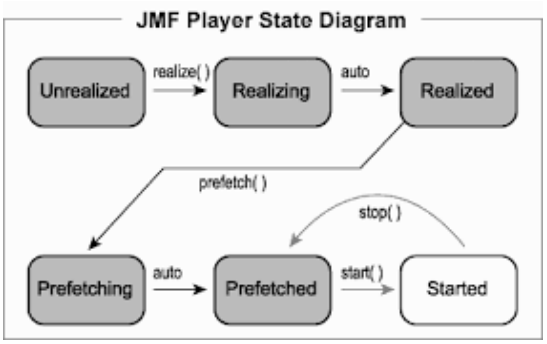


Figura 5-1: Diagrama de estados Player JMF

La descripción de los estados del objeto Player es la siguiente:

- **Unrealized:** Estado inicial. En este estado el objeto Player no conoce absolutamente nada de los datos multimedia que va a procesar.

- **Realizing:** En este momento el objeto Player determina y carga los recursos necesarios que va a necesitar para la reproducción de los datos multimedia.

- **Realized:** Aquí el objeto Player ya ha cargado con todos los recursos necesarios y conoce el tipo datos que va a reproducir.

- **Prefetching:** En este caso el objeto Player se prepara comenzar a reproducir los datos multimedia cargados.

- **Prefetched:** En este estado el Player ya está preparado para comenzar a reproducir los datos multimedia.

- **Started:** Estado que se encuentra el Player en el momento que comienza a reproducir los datos multimedia. A este estado se accede al llamar al método "Start".

En nuestra aplicación en vez de llamar a "CreatePlayer" llamamos a "CreateRealizedPlayer" para crear el objeto Player ya cargado en el estado "Realized". Desde este estado podemos ya llamar al método "Start" para comenzar la reproducción de los datos multimedia. A este método se le puede llamar desde cualquier estado, es decir, si nos encontramos en estado realized, al llamar a "Start" pasaremos por los estados "Prefetching", "Prefetched" y "Started".

Una vez creado el objeto Player serán cargadas otras clases que nos van a permitir trabajar de una manera más fácil en el manejo del vídeo. Estas clases son las siguientes:

- **FrameGrabbingControl:** Permite grabar en un buffer los datos del frame actual en el que se encuentra el vídeo.

- **FramePositioningControl:** Permite movernos a un frame específico del vídeo, así como obtener el frame a partir del segundo de reproducción del vídeo y viceversa.

- **FrameRateControl:** Permite establecer el frame rate del vídeo. Esta funcionalidad será sobretodo utilizada en la sincronización de los vídeos para evitar desfases durante la reproducción de múltiples vídeos.

Una de las dificultades que se presentaron en la carga de las clases "FrameGrabbingControl", "FramePositioningControl" y "FrameRateControl" ya que debido al formato de algunos de los vídeos que se estaban utilizando hasta este momento en las pruebas no eran compatibles. Este problema había que solucionarlo debido a que estas clases eran totalmente necesarias para la correcta realización del proyecto. Los motivos son los siguientes:

- Si no utilizábamos la clase "FramePositioningControl" para cambiar el frame del vídeo en el momento de cambiar el instante de reproducción del vídeo y estando el vídeo pausado no refrescaba la imagen del frame actual por lo que daba la sensación de que seguíamos en el mismo instante de reproducción anterior.

- Para la sincronización de los vídeos se necesitaba establecer el mismo frame rate en cada uno de ellos y sin la clase "FrameRateControl" no lo podíamos realizar.

Debido a que el problema provenía del formato del vídeo se decidió utilizar la librería FFMPEG para convertir los vídeos al formato .MOV el cual es totalmente compatible con JMF y además ofrece la misma calidad de audio y de vídeo que el vídeo original. En el *Anexo A: Librería FFMPEG* se puede ver cómo realizar la conversión de los vídeos.

Para la reproducción y pausa de los vídeos se utilizan los métodos "Start" y "Stop" del objeto Player. El método "Stop" pasa el objeto Player del estado "Started" al estado "Prefetched".

El modelo de tiempo implementado por JMF es el siguiente *(Figura 5-2)*:

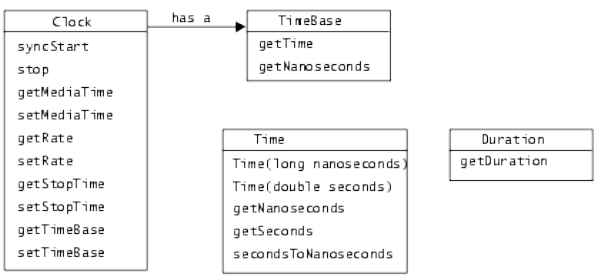


Figura 5-2: Modelo tiempo JMF

El objeto "Player" implementa la interfaz "Clock", que define las operaciones básicas de temporización y sincronización que son necesarias para poder reproducir los datos multimedia. "Clock" utiliza "timeBase" para realizar un seguimiento del paso del tiempo

mientras se están reproduciendo los datos multimedia. El tiempo "timeBase" nunca puede ser parado o reseteado. A través del objeto "Player" podemos obtener el "timeBase" llamando al método "getTimeBase" o cambiarlo llamando a "setTimeBase", en este último caso se podrá cambiar en caso de que el "timeBase" del flujo de datos multimedia soporte el nuevo "timeBase".

El objeto "mediaTime" representa el instante actual de reproducción del vídeo. El instante inicial del vídeo viene representado por el "mediaTime" cero y el final del vídeo por el máximo "mediaTime" del este. La duración del vídeo que la podemos obtener llamando al método "getDuration" que es la longitud de tiempo que se tarda en reproducir los datos multimedia.

Durante el desarrollo de la aplicación surgió un error bloqueante y es que no se actualizaba el frame actual del vídeo tras cambiar el instante actual de reproducción del vídeo, se visualizaba el frame anterior. Para solucionar esto tuvimos que utilizar la clase "FramePositioningControl" para colocarnos en un determinado frame obtenido a partir del nuevo instante de reproducción del vídeo, actualizándose visualmente de esta manera el frame. Para realizar esto se ha implementado el método "setFrameFromTime" *(Figura 5-3)* que a partir de un "mediaTime" colocamos el vídeo en el frame establecido a ese instante.

```
public void setFrameFromTime(Time time)
{
   fpc.seek(fpc.mapTimeToFrame(time));
}
```

Figura 5-3: Método setFrameFromTime

En la siguiente imagen *(Figura 5-4)* podemos ver que para cambiar el instante actual de reproducción del vídeo actualizamos el "mediaTime" y el frame del vídeo:

```
ventanaVideo.getVideo().setFrameFromTime(new javax.media.Time((double)result));
ventanaVideo.getVideo().getMediaPlayer().setMediaTime(new javax.media.Time((double)result));
```

Figura 5-4: Cambio instante reproducción vídeo

JMF API nos permite cambiar el nivel de audio de los datos multimedia que se están reproduciendo, así como silenciar o activar el audio. Para ello deberemos acceder a la clase "GainControl" *(Figura 5-5)* a partir del objeto Player, para posteriormente llamar al método "setMute" para silenciar o activar el audio. A la clase "GainControl" se le puede añadir un "Listener" para los cambios de volumen.

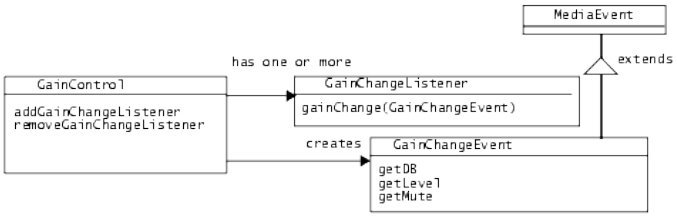


Figura 5-5: GainControl

5.1.2 Sincronización vídeos

La sincronización de los vídeos es un aspecto importante en la aplicación. Durante la reproducción múltiple de vídeos no se debe de producir un retraso o desincronización entre ellos en caso de que se hayan sincronizado ya que podría provocar que el desfase entre ellos sea notorio para el usuario impidiendo el correcto análisis de los vídeos.

Como punto de partida, comentar que finalmente no se ha utilizado la funcionalidad que ofrece JMF para la sincronización de los vídeos. Esto es debido a que no es posible marcar diferentes instantes de tiempo en cada vídeo durante la sincronización, el cual es un requisito establecido en la aplicación. Si se cambia el segundo de reproducción del vídeo maestro los vídeos esclavos reflejan este cambio marcando como segundo actual el mismo que el del vídeo maestro. Por lo tanto, la sincronización entre dos o más vídeos se realizará de forma manual tal y como se describe en este mismo punto.

Para poder sincronizar un vídeo con el vídeo maestro bastará con pulsar el botón de sincronizar *(Figura 5-6)* del panel de control correspondiente al vídeo que se quiere sincronizar.

Figura 5-6: Botón sincronizar

El botón para sincronizar sólo aparecerá en los vídeos esclavos, el vídeo uno siempre será el vídeo maestro. Una vez que hayamos sincronizado el vídeo en el panel de control veremos la siguiente imagen *(Figura 5-7)*:

Figura 5-7: Botón sincronizar (2)

Podremos sincronizar y desincronizar el vídeo al vídeo maestro tantas veces como queramos.

Cada vez que pulsemos en este botón se lanzará un evento *(Figura 5-8)* que comprobará si el vídeo es o no esclavo del vídeo maestro. En caso de que no lo sea se añadirá a la lista de vídeos esclavos, se marcará como vídeo esclavo, se deshabilitarán los controles de pausar, reproducir y el slider correspondientes al vídeo y se establecerá el frame rate y timeBase del vídeo al del vídeo maestro para que la sincronización entre ambos sea correcta y no se produzcan retrasos o desincronización entre ellos. Si ya es un vídeo esclavo se realizarán las acciones comentadas a la inversa.

```
if (!video.isIsSlave())
{
    ventanaVideo.getContVideo().setFrameRateyTimeBaseVideo(DatosControladores.getMasterVentanaVideo()
            .getContVideo().getFrameRateVideo(), DatosControladores.getMasterVentanaVideo()
                    .getContVideo().getTimeBaseVideo());

    DatosControladores.getSlavePlayers().add(ventanaVideo);

    video.setSliderValueBlocked(ventanaVideo.getSliderTime().getValue() - DatosControladores
            .getMasterVentanaVideo().getContVideo().getVideo().getSliderValueBeforeEvent());

    video.setIsSlave(true);
    try {
        img = ImageIO.read(getClass().getResource("/TFG/resources/candado_cerrado.png"));
    } catch (IOException ex) {
        Logger.getLogger(TFG.class.getName()).log(Level.SEVERE, null, ex);
    }
    buttonSlave.setIcon(new ImageIcon(img));

    // Deshabilitamos botones
    ventanaVideo.getButtonPlayPause().setEnabled(false);
    ventanaVideo.getSliderTime().setEnabled(false);
}
```

Figura 5-8: Evento sincronizar/desincronizar vídeo

En nuestra aplicación el vídeo con id uno siempre será el vídeo maestro, los siguientes vídeos añadidos serán los vídeos esclavos en caso de que el usuario quiera sincronizarlos con el vídeo maestro. Se pueden sincronizar y reproducir tantos vídeos como el usuario quiera, pero por cuestiones de rendimiento se recomienda como tope en diez la reproducción simultánea de vídeos. En el momento en el que se sincronice un vídeo esclavo al vídeo maestro se guardará la diferencia entre el instante de tiempo del vídeo maestro y el instante de tiempo del vídeo esclavo. De esta forma cada vez que se cambie el instante de reproducción del vídeo maestro el vídeo esclavo también se verá modificado en este aspecto, pero manteniendo consecuente la sincronización marcada en el inicio.

En caso de que uno de los vídeos esclavos se encuentre en un punto en el que el desfase con el vídeo maestro le provoque estar pausado en el instante cero se quedará pausado hasta que el desfase le permita comenzar su reproducción pudiendo de esta manera mantener la sincronización configurada entre ambos. Esto se hará automáticamente en el hilo de control (*ControlThread*) que se lanza por cada vídeo y se mantiene activo hasta que se cierre la aplicación o se cierre la ventana del vídeo. El hilo "ControlThread" también se encarga de actualizar el valor del slider y el segundo actual del vídeo que se muestra al usuario. En la siguiente imagen podemos ver que entre el vídeo maestro y el vídeo esclavo hay un desfase de 20 segundos y por ello, al estar el vídeo maestro en el segundo dieciséis el vídeo esclavo se encuentra en el instante cero y en pausa *(Figura 5-9)*:

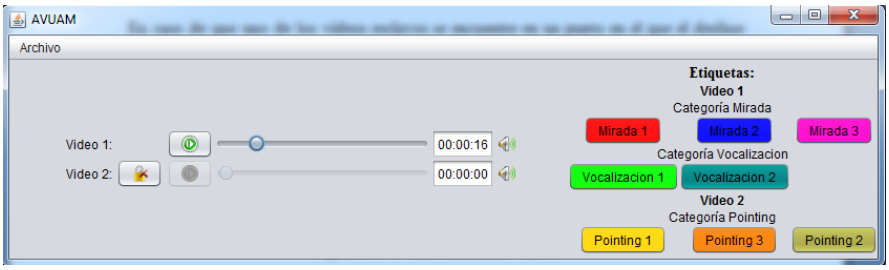


Figura 5-9: Sincronización vídeos

En el momento en el que el vídeo maestro se encuentre en el segundo veinte el vídeo esclavo comenzará a reproducirse tal y como podemos ver en la siguiente imagen *(Figura 5-10)*:

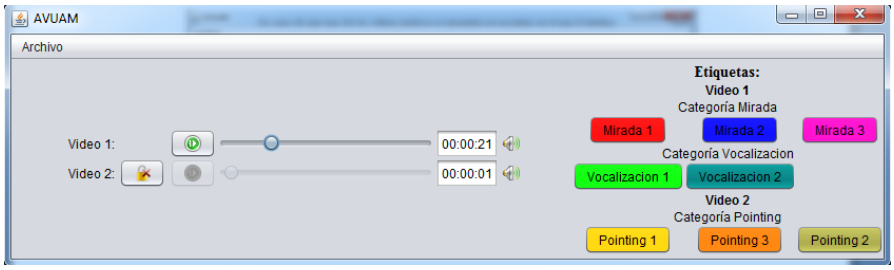


Figura 5-10: Sincronización vídeos (2)

Para realizar esta comprobación en el hilo de control *(Figura 5-11)* se mira que sea un vídeo esclavo, esté en pausa y el vídeo maestro reproduciéndose y que el desfase respecto al vídeo maestro más el segundo de reproducción del vídeo maestro sea mayor igual que cero. Si se cumplen estas condiciones se comienza a reproducir el vídeo esclavo.

```
if (contVideo.getVideo().isIsSlave())
{
    if (contVideo.getVideo().isIsPaused() && !DatosControladores.getMasterVentanaVideo()
            .getContVideo().getVideo().isIsPaused())
    {
        if (contVideo.getVentanaVideo().getSliderTime().getValue() == contVideo.getVentanaVideo()
                .getSliderTime().getMinimum())
        {
            if ((contVideo.getVideo().getSliderValueBlocked() + DatosControladores.getMasterVentanaVideo()
                    .getContVideo().getSegundoActualVideo())
                >= contVideo.getVentanaVideo().getSliderTime().getMinimum() && !DatosControladores
                        .getMasterVentanaVideo().getContVideo().getVideo().isIsMasterVideoChanged())
            {
                contVideo.getVideo().setPausedBySync(false);
                contVideo.playPauseVideo();
            }
        }
    }
}

// Actualizar slider y caja de texto con segundo actual del video
SimpleDateFormat format = new SimpleDateFormat("HH:mm:ss");
Calendar time = SecondsToCalendar((int)contVideo.getSegundoActualVideo());

contVideo.getVentanaVideo().getFieldTiempo().setValue(format.format(time.getTime()));
contVideo.getVentanaVideo().getSliderTime().setValue((int)contVideo.getSegundoActualVideo());
contVideo.getVideo().setSliderValueBeforeEvent((int)contVideo.getSegundoActualVideo());
```

Figura 5-11: Hilo "ControlThread"

En el momento en que se cambia el instante de reproducción del vídeo maestro a través del slider se lanza un hilo por cada vídeo esclavo en el que se calcula el punto en el que se tiene que colocar cada vídeo esclavo. Este cálculo no hace falta hacerlo en el vídeo maestro ya que se coloca directamente sobre el instante en el que se pulse en el slider. Se lanza un hilo para realizar este cálculo por cada vídeo para evitar que se produzcan retrasos en la sincronización entre el vídeo maestro y los vídeos esclavos, ya que un cálculo secuencial podría provocar que el último vídeo o últimos vídeos, en un entorno en el que se están reproduciendo bastantes vídeos sincronizados, tuviese un retraso en la sincronización configurada inicialmente. En el cálculo del vídeo, dependiendo del punto en el que se haya colocado el slider del vídeo maestro, se podrán distinguir tres casos:

- El desfase configurado entre el vídeo esclavo y el vídeo maestro más el nuevo instante de reproducción del vídeo maestro sea menor o igual que cero: En este caso el vídeo esclavo se quedará en el instante cero de reproducción.

- El desfase configurado entre el vídeo esclavo y el vídeo maestro más el nuevo instante de reproducción del vídeo maestro sea mayor o igual que la duración del vídeo esclavo: En este caso el vídeo esclavo se quedará en el instante final de reproducción.

- Si no se cumple ninguno de los dos anteriores casos es que el instante de reproducción es el desfase entre el vídeo esclavo y el maestro mas el instante de reproducción del vídeo maestro.

En la siguiente imagen *(Figura 5-12)* podemos ver el código del hilo que realiza el cálculo comentado por cada vídeo esclavo cada vez que se cambia el instante de reproducción del vídeo maestro:

```
contVideo.pausarVideo();
if ((contVideo.getVideo().getSliderValueBlocked() + result) >= contVideo.getVentanaVideo().getSliderTime()
        .getMaximum())
{
    contVideo.getVideo().setPausedBySync(true);
    contVideo.setFrameFromTimeVideo(new javax.media.Time(contVideo.getDuracion()));
    contVideo.setMediaTimeVideo(new javax.media.Time(contVideo.getDuracion()));
}
else if ((contVideo.getVideo().getSliderValueBlocked() + result) <= contVideo.getVentanaVideo().getSliderTime()
        .getMinimum())
{
    contVideo.getVideo().setPausedBySync(true);
    contVideo.setFrameFromTimeVideo(new javax.media.Time((double)0));
    contVideo.setMediaTimeVideo(new javax.media.Time((double)0));
}
else
{
    if (contVideo.getVideo().isPausedBySync() || !DatosControladores.getMasterVentanaVideo().getContVideo()
            .getVideo().isIsPaused())
    {
        contVideo.getVideo().setPausedBySync(false);
    }
    contVideo.setFrameFromTimeVideo(new javax.media.Time((double)contVideo.getVideo().getSliderValueBlocked()
            + (double)result));
    contVideo.setMediaTimeVideo(new javax.media.Time((double)contVideo.getVideo().getSliderValueBlocked()
            + (double)result));
}
```

Figura 5-12: Hilo "ActionMouseSliderThread"

Cada vez que pausemos o reproduzcamos el vídeo maestro se tienen que pausar o reproducir respectivamente todos los vídeos esclavos. Por ello, y para que no produzca ningún retraso al realizar esta acción por cada vídeo esclavo, se lanza un hilo *(Figura 5-13)* por cada vídeo esclavo que se encargará de pausar o reproducir este. El hilo llama a los métodos "Start" y "Stop" de la clase Player de JMF.

```
if (!contVideo.getVideo().isPausedBySync())
{
    if (contVideo.getVideo().isIsPaused())
    {
        contVideo.reproducirVideo();
    }
    else
    {
        contVideo.pausarVideo();
    }
}
```

Figura 5-13: Hilo "ActionMousePlayPauseThread"

En esta última imagen se puede ver que el vídeo sólo se reproducirá o pausará en caso que de que la variable "PausedBySync" sea igual a false. Esto es debido a que si tras cambiar el instante de reproducción del vídeo maestro a través del slider el vídeo esclavo se coloca en

el instante cero de reproducción y el vídeo maestro continúa reproduciéndose, evitar que si se para el vídeo maestro se empiece a reproducir el vídeo esclavo al lanzarse el hilo. Esta variable se pone a true en caso que tenga que estar pausado el vídeo esclavo tras el cálculo lanzado por el hilo "*ActionMouseSliderThread*" y se pone a false en caso de que el vídeo pueda comenzar a reproducirse tras cumplirse la condición del hilo "*ControlThread*" o se cumpla la tercera condición indicada anteriormente para el hilo "*ActionMouseSliderThread*".

5.1.3 Etiquetas

Durante la reproducción de uno o más vídeos el usuario puede aplicar etiquetas manualmente. Para ello el usuario pulsará sobre la etiqueta que quiera aplicar y se quedará pulsando en el transcurso de tiempo en el que la quiera aplicar. Esta etiqueta se guardará en una lista de etiquetas y será presentada en el panel de control en el rango de tiempo que haya sido aplicada. Si el usuario mantiene el cursor sobre una de las etiquetas aplicadas aparecerá un tooltip *(Figura 5-14)* que indicará el instante inicial y final en el que se ha aplicado la etiqueta, así como la descripción de esta.

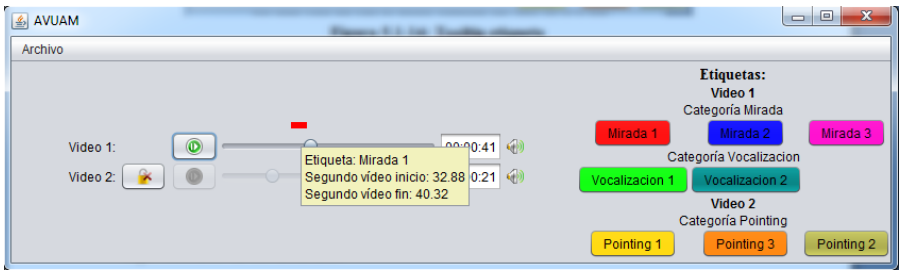


Figura 5-14: Tooltip etiqueta

Un punto a indicar es que las etiquetas aplicadas a los vídeos esclavos siempre serán pintadas en el eje "x" en la posición inicial y final del segundo de reproducción del vídeo maestro. Los segundos iniciales y finales almacenados en la etiqueta serán los segundos del vídeo esclavo. Esto se hace así porque, aunque el evento ocurra en el vídeo esclavo se utiliza como referencia a la hora de pintar la etiqueta el instante de reproducción del vídeo maestro para evitar dar lugar a confusiones al usuario. En caso de que se tengan etiquetas configuradas para el vídeo dos y luego en el momento del análisis sólo se abra un vídeo las etiquetas que se apliquen y que estén asignadas al vídeo dos no se pintarán ni se aplicarán. Esto mismo ocurre para todas las etiquetas configuradas para vídeos que no se estén reproduciendo en ese momento.

En el momento en el que el usuario pulsa sobre el botón de una etiqueta se crea una etiqueta que se inicializa con el instante actual de reproducción del vídeo, la posición en el eje "x" y un id único, posteriormente se añade a la lista de etiquetas almacenada en la aplicación y se representa en el panel de control. Finalmente se lanza un hilo *(Figura 5-16)* que se ejecutará mientras el usuario mantenga pulsado el botón de la etiqueta. Este hilo actualizará el instante final en el que se aplica la etiqueta, aumentará el tamaño horizontal de esta acorde al segundo último que se esté aplicando y además calculará su posición en el eje "y" para evitar que se solape con otra etiqueta ya aplicada. Una vez el usuario suelte el botón el hilo actualizará el tooltip de la etiqueta y actualizará la posición en el eje "y" de esta. En la figura 5-15 podemos ver varias etiquetas apiladas:

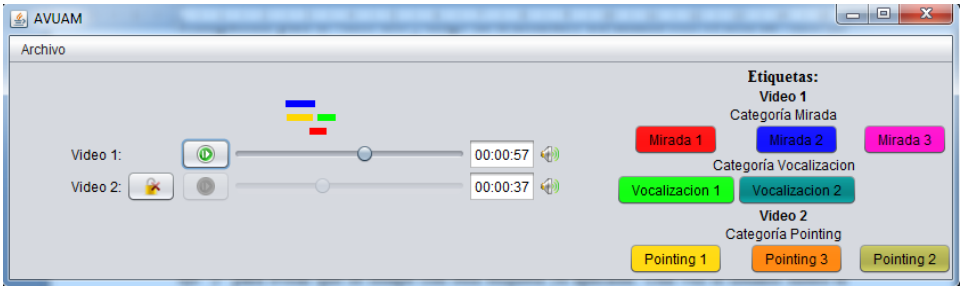


Figura 5-15: Etiquetas apiladas eje "y"

La posición en el eje "x" se calcula de la siguiente manera:

- Se obtiene el ancho del slider.
- Se obtiene la duración en segundos del vídeo.
- Se divide el ancho del slider entre la duración del vídeo en segundos para obtener el número de pixels que hay para cada segundo en el slider.
- Se multiplica el número de pixels por segundo por el instante actual de reproducción del vídeo en segundos. El resultado de esta operación será la actual posición final de la etiqueta en el eje "x".

Para calcular la posición en el eje "y" de la etiqueta comprobamos si en la posición "x" final y la posición "y" actual hay algún componente que sea de tipo "Etiqueta", si es así, se resta a la posición "y" actual el alto de la etiqueta en bucle hasta comprobar que en la posición "y" no existe ningún componente. Una vez cumplida la condición se sale del bucle y se pinta la etiqueta en la nueva posición en el eje "y".

```
while (DatosControladores.isMousePressed())
{
    etiqueta.setFinalSecond(contVideo.getSegundoActualVideo());

    int sliderWidth = DatosControladores.getMasterVentanaVideo().getSliderTime().getWidth();
    double videoDuration = DatosControladores.getMasterVentanaVideo().getContVideo().getDuracion();
    double pixelPerSecond = sliderWidth / videoDuration;
    double posInicial = etiqueta.getxPos();
    double posFinal = pixelPerSecond * DatosControladores.getMasterVentanaVideo().getContVideo()
            .getSegundoActualVideo();
    Dimension sizeEtiqueta = etiqueta.getPreferredSize();

    while (analizePanel.getComponentAt(new Point((int)(DatosControladores.getMasterVentanaVideo().getSliderTime()
            .getX() +
            posFinal + 2), posY)) != null
            && analizePanel.getComponentAt(new Point((int)(DatosControladores.getMasterVentanaVideo().getSliderTime()
                    .getX() + posFinal + 2), posY)) instanceof EtiquetaButton)
    {
        posY = posY - sizeEtiqueta.height;
    }

    etiqueta.setBounds((int)(DatosControladores.getMasterVentanaVideo().getSliderTime().getX() + posInicial), posY,
            (int)posFinal - (int)posInicial + 2, sizeEtiqueta.height);

    analizePanel.repaint();
}

etiqueta.setyPos(posY);
etiqueta.actualizaToolTipText();
```

Figura 5-16: Hilo EtiquetaThread

Si el usuario se equivoca al aplicar una etiqueta podrá eliminar la etiqueta ya aplicada o cambiarla por otra. Para ello deberá dar click derecho sobre la etiqueta a cambiar/eliminar para que aparezca un menú en el que podrá elegir cualquiera de las dos opciones mencionadas. Para realizar esto se ha creado un menú dinámico al que se le van añadiendo todas las etiquetas almacenadas en el fichero de configuración para el submenú de cambiar

etiqueta. Este menú se desplegará al dar click derecho sobre una etiqueta ya aplicada y en la posición "x" e "y" sobre la que se haya pulsado y dentro del rango de una etiqueta.

5.1.4 Lectura fichero configuración XML

En cualquier momento el usuario podrá cargar un fichero de configuración XML con las etiquetas configuradas en el mismo. Tras realizar esto el usuario visualizará en el panel derecho del panel de control las etiquetas que se hayan configurado en este XML divididas por el vídeo al que se hayan asociado.

Para la lectura de los archivos XML se ha utilizado la librería SAX. SAX en una API para XML basado en eventos. Es decir, va recorriendo el XML desde el inicio y va generando eventos conforme va encontrando las diferentes etiquetas del XML. En los eventos generados en estas etiquetas será donde nosotros recojamos la información que necesitemos recuperar.

En el evento que se lanza cuando encuentra una etiqueta de inicio, "startElement" *(Figura 5-17)*, recuperaremos el atributo "Id" del vídeo cuando la etiqueta de apertura sea "Video" y crearemos un objeto "XMLButton" pasándole como argumento la categoría obtenida a partir del atributo de esta etiqueta de apertura y el id del vídeo recuperado anteriormente cuando la etiqueta de apertura sea "Button".

```
@Override
public void startElement(String uri, String localName, String qName,
    Attributes attributes) throws SAXException {

    // Etiqueta video
    if (qName.equalsIgnoreCase("VIDEO"))
    {
        idVideo = attributes.getValue("id");
    }
    // Etiqueta button
    if (qName.equalsIgnoreCase("BUTTON"))
    {
        categoria = attributes.getValue("category");
        xmlButton = new XMLButton(idVideo, categoria);
    }
    // Etiqueta id
    if (qName.equalsIgnoreCase("ID"))
    {
        bId = true;
    }
    // Etiqueta description
    if (qName.equalsIgnoreCase("DESCRIPTION"))
    {
        bDescription = true;
    }
    // Etiqueta color
    if (qName.equalsIgnoreCase("COLOR"))
    {
        bColor = true;
    }
}
```

Figura 5-17: Evento startElement API SAX

Para el evento "endElement" *(Figura 5-18)*, que se lanza al encontrar una etiqueta de cierre, de la etiqueta "Button" se añadirá el objeto "XMLButton" a la lista de etiquetas XML de la clase.

```
@Override
public void endElement(String uri, String localName,
    String qName) throws SAXException {

    if (qName.equalsIgnoreCase("BUTTON"))
    {
        XMLButtons.add(xmlButton);
    }
}
```

Figura 5-18: Evento endElement API SAX

En el evento "characters" *(Figura 5-19)*, lanzado cada vez que se encuentra una cadena de texto, recogeremos la información de las etiquetas XML. En la siguiente imagen podemos ver como se recuperan los diferentes contenidos de las etiquetas XML: id, descripción y color en formato RGB.

```
@Override
public void characters(char ch[], int start, int length) throws SAXException
{
    // Etiqueta id
    if (bId)
    {
        xmlButton.setId(new String(ch, start, length));
        for (XMLButton button : XMLButtons)
        {
            if (button.getIdVideo().equalsIgnoreCase(idVideo)
                    && button.getId().equalsIgnoreCase(xmlButton.getId()))
            {
                throw new SAXException("Formato del XML incorrecto.");
            }
        }
        bId = false;
    }
    // Etiqueta description
    if (bDescription)
    {
        xmlButton.setText(new String(ch, start, length));
        xmlButton.setDescription(new String(ch, start, length));
        bDescription = false;
    }
    // Etiqueta color
    if (bColor)
    {
        String color = new String(ch, start, length);
        String[] rgb = color.split(",");
        xmlButton.setColor(new Color(Integer.parseInt(rgb[0]), Integer.parseInt(rgb[1]),
                Integer.parseInt(rgb[2])));
        bColor = false;
    }
}
```

Figura 5-19: Evento characters API SAX

5.1.5 Exportación datos

La aplicación permitirá en cualquier momento exportar los datos que se han recopilado hasta ese momento. En el momento de realizar esta exportación se da la posibilidad de elegir el nombre y ubicación del fichero que almacenará todos estos datos. En caso de que exista ya un fichero con este nombre aparecerá un aviso de si se quiere sobrescribir el fichero anterior *(Figura 5-20)*.

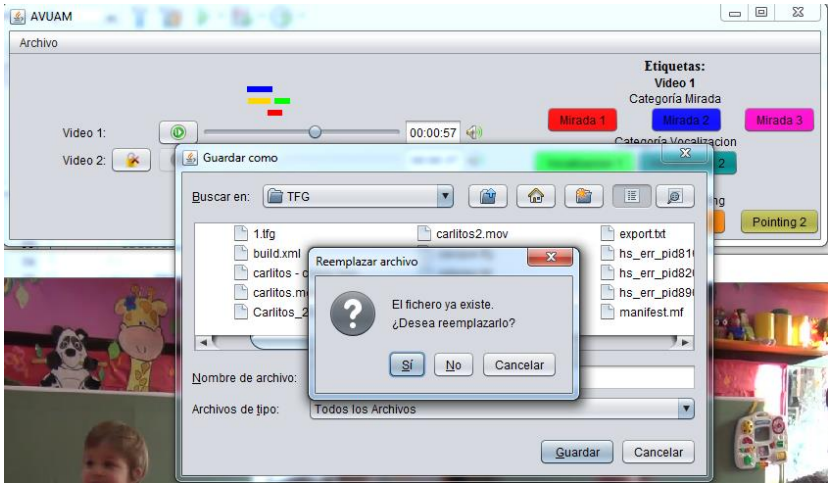


Figura 5-20: Exportación datos en fichero

Para la exportación de estos datos se han contemplado dos formatos: texto separado por comas y texto tabulado. Ambos formatos de archivo se guardarán en formato .csv para facilitar su posterior lectura y análisis.

Las columnas a incluir en el fichero de exportación serán las siguientes *(Figura 5-21)*:

Id evento	Id etiqueta	Descripción etiqueta	Video	Desfase	Instante inicial	Instante final	Instante inicial con desfase	Instante final con desfase	Duración total evento
666f77d0-724c-4d28-874c-c7dc6366fdf8	1	Mirada 1	1	0	32.07	39.96	32.07	39.96	7.89
a16e3a3c-0a22-4b99-b87b-d6abf2014acf	5	Vocalizacion 2	1	0	41.88	48.61	41.88	48.61	6.73
aca75de9-e617-4427-a9a8-42acd214b88f	3	Mirada 2	1	0	71.79	78.94	71.79	78.94	7.15
47b3d32d-71bd-4b19-b353-fa5cc9a9bec4	2	Pointing 3	2	20	62.89	68.66	82.89	88.66	5.77
cec2ca92-bc15-4742-a352-7c55b4df7fd6	2	Vocalizacion 1	1	0	24.6	33.51	24.60	33.51	8.91
94d6eb01-ce8c-4b58-93b8-df7883d8ee4d	1	Pointing 1	2	20	16.16	27.87	36.16	47.87	11.71

Figura 5-21: Formato fichero exportación con comas

- **Id Evento:** Identificador único del evento.
- **Id Etiqueta:** Identificador de la etiqueta. Dos etiquetas de distintos vídeos pueden tener el mismo id.
- **Descripción etiqueta:** Descripción de la etiqueta configurable por el usuario.
- **Vídeo:** Id del vídeo al que se asignó la etiqueta.
- **Desfase:** Si el vídeo estaba sincronizado este valor indica la diferencia en el instante de tiempo entre el vídeo maestro y el mismo.
- **Instante inicial:** Instante de tiempo en el que se comenzó a aplicar la etiqueta.
- **Instante final:** Instante de tiempo en el que se dejó de aplicar la etiqueta.
- **Instante inicial con desfase:** Instante de tiempo en el que se comenzó a aplicar la etiqueta mas el desfase configurado respecto al vídeo maestro. En caso de ser una etiqueta que pertenece al vídeo maestro el tiempo es el mismo que para la columna "Instante inicial".

- **Instante final con desfase:** Instante de tiempo en el que se dejó de aplicar la etiqueta mas el desfase configurado respecto al vídeo maestro. En caso de ser una etiqueta que pertenece al vídeo maestro el tiempo es el mismo que para la columna "Instante final".

- **Duración total evento:** Diferencia entre las columnas instante final e instante inicial.

5.1.6 Carga y generación de plantillas

Un punto a indicar, es que el usuario en cualquier momento puede guardar el estado actual de la anotación de vídeos realizada, pudiendo recuperar toda esta información posteriormente para proseguir con el análisis. Esto facilita mucho el trabajo al usuario en situaciones en las que el análisis de los vídeos pueda llevar más de un día o trabaje con los mismos vídeos en varias ocasiones a lo largo de un mismo día.
En el momento de guardar la plantilla se da la posibilidad de elegir el nombre y ubicación de esta. En caso de que exista ya una plantilla con este nombre aparecerá un aviso de si se quiere sobrescribir la plantilla anterior *(Figura 5-20)*. Las plantillas guardadas se almacenarán en un fichero de texto con la estructura que se muestran en las figuras 5-22 y 5-23. A estos ficheros de texto se les ha puesto la extensión .tfg.

La plantilla guardada contendrá los siguientes datos:

- Una cabecera con la información de los vídeos reproducidos:

```
Video, Ruta video, Instante, Sincronizado, Desfase
1, C:\Users\Luisfer\Documents\NetBeansProjects\TFG\carlitos.mov, 45.47, false, 0
2, C:\Users\Luisfer\Documents\NetBeansProjects\TFG\carlitos2.mov, 25.43, true, -20
```

Figura 5-22: Cabecera plantilla

  a. **Vídeo:** Id del vídeo asignado.

  b. **Ruta vídeo:** Ruta donde está el vídeo almacenado.

  c. **Instante:** Segundo de reproducción del vídeo en el que se dejó.

  d. **Sincronizado:** En caso de que no sea el vídeo maestro será un valor booleano que indicará si el vídeo está sincronizado o no.

  e. **Desfase:** Si el vídeo estaba sincronizado este valor indica la diferencia en el instante de tiempo entre el vídeo maestro y el propio vídeo esclavo.

- Lista de etiquetas aplicadas por el usuario:

```
Id evento, Id etiqueta, Video, Instante inicial, Instante final, PosY
b84e5158-d018-426d-9963-490f1f0cd92a, 1, 1, 30.54, 34.55, 1
a313d1fa-2c63-40b7-beb8-780e9cea5105, 2, 2, 16.11, 18.62, 1
a7bb5319-7a53-44c3-8f2c-c9ad8fd6dd87, 3, 2, 21.19, 24.19, 1
```

Figura 5-23: Lista etiquetas plantilla

a. **Id evento:** Identificador único del evento.

b. **Id etiqueta:** Identificador de la etiqueta. Dos etiquetas de distintos vídeos pueden tener el mismo id.

c. **Vídeo:** Id del vídeo al que se asignó la etiqueta.

d. **Instante inicial:** Instante de tiempo en el que se comenzó a aplicar la etiqueta.

e. **Instante final:** Instante de tiempo en el que se dejó de aplicar la etiqueta.

f. **PosY:** Como las etiquetas pueden estar apiladas, este valor indica el número de posición en el eje de la "y".

A través de la aplicación se podrá cargar cualquier plantilla en formato .tfg. En la ventana de elección del fichero de la plantilla a cargar aparecerán solos los ficheros que tengan formato .tfg. Una vez cargada la plantilla se perderán las etiquetas aplicadas y los vídeos cargados en el anterior análisis. Por seguridad, la aplicación mostrará un mensaje de aviso en la que pedirá la confirmación del usuario.

5.2 Editor XML

Es la aplicación encargada de editar o crear nuevos XML de configuración para posteriormente ser cargados en la aplicación principal.

Para la lectura de los archivos XML se ha utilizado la misma API que en la aplicación principal, SAX, tal y como se puede ver en el punto *5.1.4 Lectura fichero configuración XML*. Una vez cargados los vídeos y las etiquetas del XML se presentan en forma de árbol en la ventana de la aplicación *(Figura 5-25)*. Como nodo raíz se encontrará el nodo "Vídeos" y de este colgarán todos los vídeos configurados. A su vez, de cada vídeo colgarán las etiquetas asignadas a este.

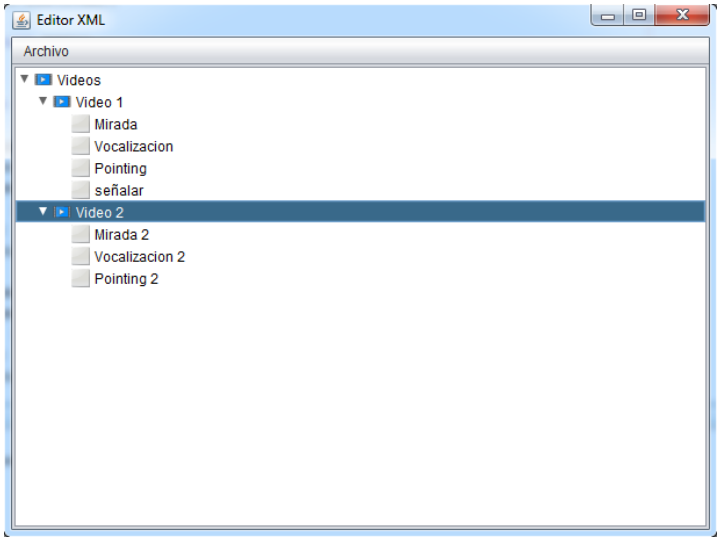


Figura 5-24: Árbol elementos Editor XML

Desde esta ventana se podrá añadir o eliminar nuevos vídeos, así como añadir, eliminar o modificar etiquetas ya existentes. Estas acciones se podrán realizar en el menú de cada nodo. El menú del nodo dependerá del nivel en el árbol en el que se ubique este.:

- Nodo raíz "Vídeos": Menú para añadir vídeos *(Figura 5-26)*.

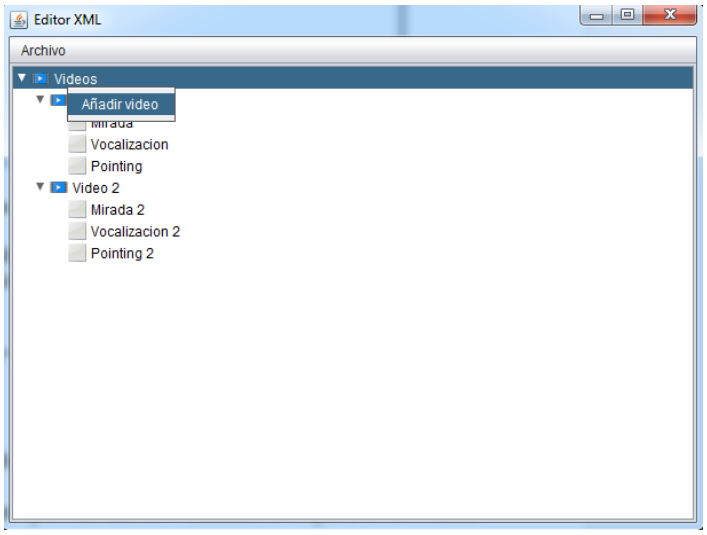


Figura 5-25: Menú nodo raíz

- Nodo "Vídeo n": Menú para eliminar vídeo o añadir nuevas etiquetas a este vídeo *(Figura 5-27)*.

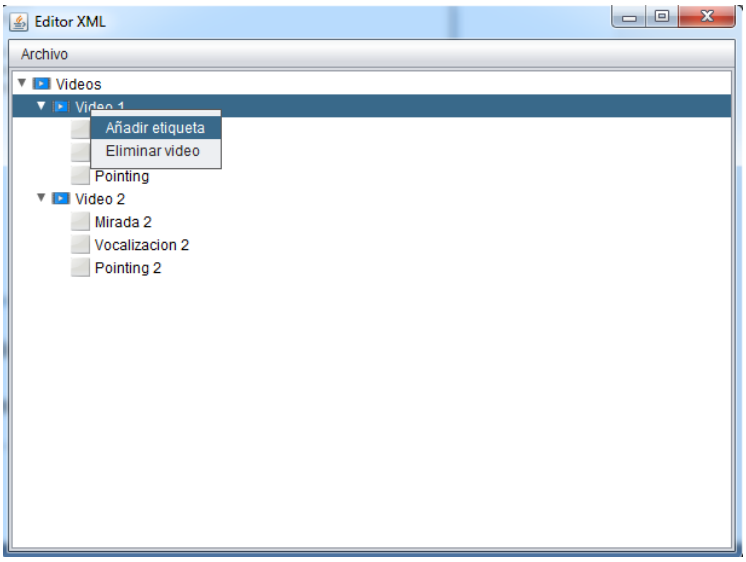


Figura 5-26: Menú nodo "Vídeo n"

- Nodo etiqueta: Menú para modificar o eliminar la etiqueta seleccionada *(Figura 5-28)*.

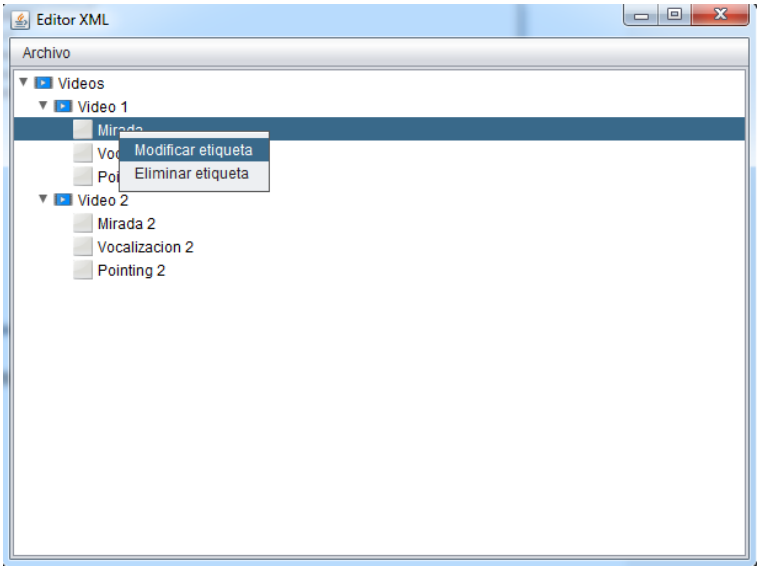


Figura 5-27: Menú nodo etiqueta

En cambio, para la modificación y generación de nuevos ficheros XML se ha utilizado la clase "XMLStreamWriter" de la API StAX (Streaming API for XML) ya que facilita en gran medida la generación de estos.

La generación de ficheros XML con esta API es sencilla, basta con utilizar los siguientes métodos:

- **WriteStartDocument:** Escribe en el fichero de salida la declaración del archivo XML, "<?xml version="1.0" ?>".

- **WriteEndDocument:** Cierra las etiquetas de apertura con su correspondiente etiqueta de cierre.

- **WriteStartElement:** Inicio de una etiqueta de apertura con el nombre pasado como argumento.

- **WriteEndElement:** Etiqueta de cierre con el nombre pasado como argumento.

- **WriteCharacters:** Escribe una cadena de caracteres entre las etiquetas de apertura y cierre en las que se llame a este método.

- **WriteAttribute:** Añade un atributo a una etiqueta de apertura. Para ello se le pasa como argumentos el nombre del atributo y el valor de este.

En la siguiente imagen *(Figura 5-29)* podemos ver cómo se generan los archivos de configuración XML en la aplicación:

```
XMLOutputFactory xMLOutputFactory = XMLOutputFactory.newInstance();
XMLStreamWriter xMLStreamWriter = xMLOutputFactory.createXMLStreamWriter(stringWriter);
xMLStreamWriter.writeStartDocument();
xMLStreamWriter.writeStartElement("TFG");

// Construccion de los datos del XML
for (XMLVideo xmlVideo : xmlVideos)
{
    xMLStreamWriter.writeStartElement("video");
    xMLStreamWriter.writeAttribute("id", String.valueOf(xmlVideo.getIdVideo()));
    for (XMLButton xmlButton : xmlVideo.getXmlButtons())
    {
        xMLStreamWriter.writeStartElement("button");
        xMLStreamWriter.writeAttribute("category", xmlButton.getCategory());
        // Id
        xMLStreamWriter.writeStartElement("id");
        xMLStreamWriter.writeCharacters(xmlButton.getId());
        xMLStreamWriter.writeEndElement();
        // Descripcion
        xMLStreamWriter.writeStartElement("description");
        xMLStreamWriter.writeCharacters(xmlButton.getDescription());
        xMLStreamWriter.writeEndElement();
        // Color
        xMLStreamWriter.writeStartElement("color");
        String color = String.valueOf(xmlButton.getColor().getRed()) + "," +
                String.valueOf(xmlButton.getColor().getGreen())
                + "," + String.valueOf(xmlButton.getColor().getBlue());
        xMLStreamWriter.writeCharacters(color);
        xMLStreamWriter.writeEndElement();

        xMLStreamWriter.writeEndElement();
    }
    xMLStreamWriter.writeEndElement();
}
xMLStreamWriter.writeEndElement();
xMLStreamWriter.writeEndDocument();
```

Figura 5-28: Generación archivos XML

5.3 Archivo XML

Es el fichero que contiene toda la configuración de las etiquetas. El formato que se ha elegido es el siguiente *(Figura 5-30)*:

```xml
<?xml version="1.0" ?>
<TFG>
    <video id="1">
        <button category="Mirada">
            <id>1</id>
            <description>Mirada 1</description>
            <color>255,0,0</color>
        </button>
        <button category="Vocalizacion">
            <id>2</id>
            <description>Vocalizacion 1</description>
            <color>0,255,0</color>
        </button>
        <button category="Mirada">
            <id>3</id>
            <description>Mirada 2</description>
            <color>0,0,255</color>
        </button>
        <button category="Mirada">
            <id>4</id>
            <description>Mirada 3</description>
            <color>255,0,204</color>
        </button>
        <button category="Vocalizacion">
            <id>5</id>
            <description>Vocalizacion 2</description>
            <color>0,102,102</color>
        </button>
    </video>
    <video id="2">
        <button category="Pointing">
            <id>1</id>
            <description>Pointing 1</description>
            <color>255,215,0</color>
        </button>
        <button category="Pointing">
            <id>2</id>
            <description>Pointing 3</description>
            <color>255,127,0</color>
        </button>
        <button category="Pointing">
            <id>3</id>
            <description>Pointing 2</description>
            <color>142,142,56</color>
        </button>
    </video>
</TFG>
```

Figura 5-29: Formato archivo XML

- **Tag "TFG":** Es el nodo raíz del XML.
    - **Tag "Vídeo":** Se configurarán tantos como vídeos se vayan a reproducir durante el análisis. El atributo "id" indica el id del vídeo al que va a estar asignado. Cada vídeo contendrá n etiquetas.
        - **Tag "Button":** Se definirá por cada etiqueta que se quiera añadir. Como atributo contiene la categoría en la que estará agrupada la etiqueta.
            - **Tag "Id":** Id de la etiqueta. Para un mismo video no se puede repetir el id de la etiqueta. En caso de que se repita el id de la etiqueta la aplicación mostrará un mensaje al leer el XML informando de que el formato del archivo XML es incorrecto.
            - **Tag "Description":** Descripción de la etiqueta configurada.
            - **Tag "Color":** Color de la etiqueta configurada.

6 Integración, pruebas y resultados

Durante el desarrollo y fin del proyecto se han realizado múltiples pruebas para comprobar el correcto funcionamiento de la aplicación. Lo que más se ha analizado en las pruebas es comprobar que no se produzca ningún retraso o desfase en la reproducción simultánea de dos o más vídeos y que las etiquetas aplicadas en un tramo de tiempo sean exactas a los instantes de tiempo en las que se querían aplicar. Otro punto importante es comprobar que los datos exportados desde la aplicación sean totalmente correctos y correspondan con las etiquetas aplicadas durante el análisis de los vídeos, además de que no haya ningún error o pérdida de precisión en los instantes de tiempo de cada etiqueta que se haya aplicado.

Para comprobar esto se han realizado diferentes pruebas. Las figuras 6-1 y 6-2 muestran un ejemplo de ellas.

Se han aplicado etiquetas durante la reproducción simultánea de dos vídeos sincronizados, el vídeo 1 en el segundo 30 y el vídeo 2 en el segundo 10. Por lo que el desfase entre ellos es de 20 segundos. Se han configurado dos categorías para el vídeo 1 y una categoría para el vídeo 2 *(Figura 6-1)*.

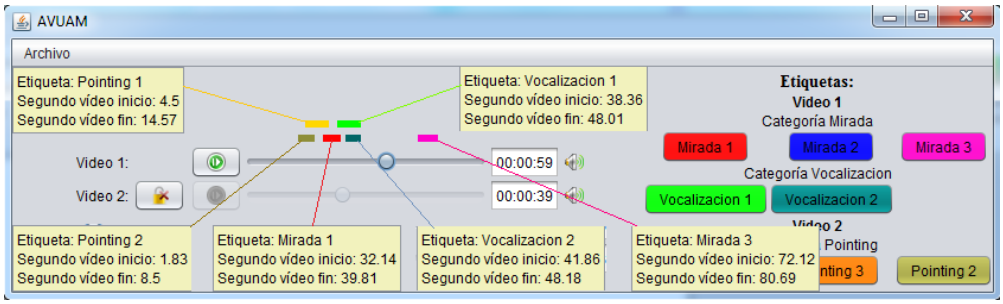


Figura 6-1: Datos etiquetas aplicadas

Ahora procedemos a comparar estos datos con los datos obtenidos en la exportación. En este caso se ha elegido exportar los datos separados por comas para posteriormente cargarlos en Excel *(Figura 6-2)*:

Id evento	Id etiqueta	Descripción etiqueta	Video	Desfase	Instante inicial	Instante final	Instante inicial con desfase	Instante final con desfase	Duración total evento
0552d534-e94e-4af3-957a-738fd512c761	1	Mirada 1	1	0	32.14	39.81	32.14	39.81	7.67
771f2528-5ef5-47f4-8af6-ee9c2027320a	5	Vocalizacion 2	1	0	41.86	48.18	41.86	48.18	6.32
ec718d97-1599-42fe-a8ca-6307dd49a6d0	1	Pointing 1	2	20	4.5	14.57	24.50	34.57	10.07
e5c826e3-d6cf-4202-a0ac-44ae92b94d5c	2	Vocalizacion 1	1	0	38.36	48.01	38.36	48.01	9.65
e1c783e9-004a-4bb6-8c7a-8433ebc8cc5f	4	Mirada 3	1	0	72.12	80.69	72.12	80.69	8.57
e4977fed-da40-460b-b0d1-55cf8c4b67de	3	Pointing 2	2	20	1.83	8.5	21.83	28.50	6.67

Figura 6-2: Exportación de los datos separados por comas

Como podemos ver, comparando las figuras *6-1* y *6-2*, los datos son correctos, no se ha perdido precisión en los instantes de tiempo inicial y final de cada etiqueta, además de ver que las etiquetas aplicadas durante el análisis se corresponden con las etiquetas que podemos ver en los datos exportados siendo los datos de cada etiqueta en ambas imágenes correctos. Durante esta prueba también se ha comprobado que las etiquetas que se han ido aplicando sean las que se estaban pulsando en ese momento y que los instantes de tiempo se correspondían con los instantes en los que se comenzaba a pulsar la etiqueta y se soltaba.

Un ejemplo de prueba a mostrar, de las diferentes pruebas realizadas, para comprobar que se mantiene la sincronización entre dos o más vídeos en los diferentes casos de uso que se pueden dar es el siguiente. Los datos iniciales son:

- Diferencia de veinte segundos entre los instantes de sincronización de los vídeos.
- La duración tanto del vídeo esclavo como del vídeo maestro será de cien segundos.
- El punto inicial de sincronización del vídeo esclavo es el segundo diez y el del vídeo maestro el segundo treinta.

a) El usuario navega al segundo diecinueve o menos del vídeo maestro y por lo tanto el vídeo esclavo se sitúa en el instante de tiempo cero. El vídeo esclavo estará en pausa hasta el momento en el que el vídeo maestro se encuentre en el segundo veinte o más. Momento en el que el vídeo esclavo automáticamente se empezará a reproducir para mantener la sincronización realizada entre ambos vídeos *(Figura 6-3)*.

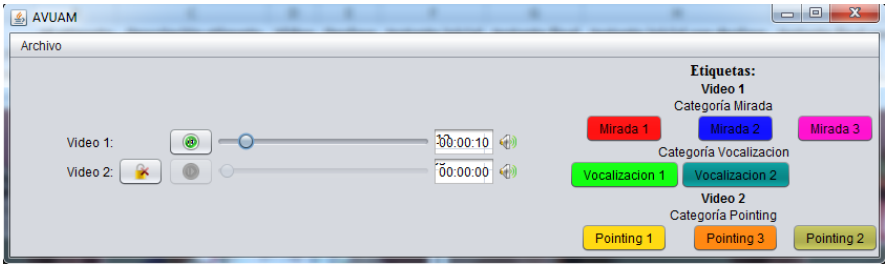


Figura 6-3: Vídeo maestro (Id 1) y vídeo esclavo (Id 2) sincronizados (3)

b) El vídeo maestro se encuentra entre los segundos veinte y cien por lo que ambos vídeos se estarán reproduciendo simultáneamente *(Figura 6-4)*.

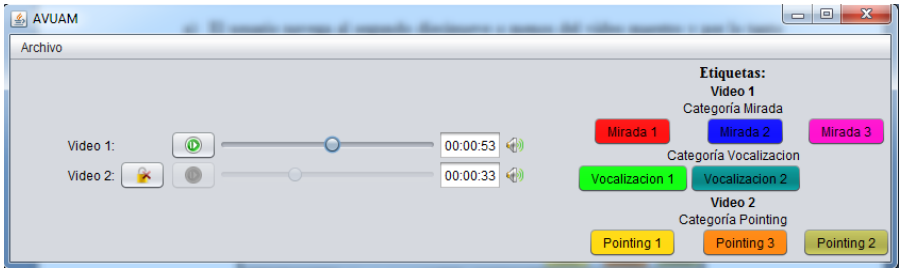


Figura 6-4: Vídeo maestro (Id 1) y vídeo esclavo (Id 2) sincronizados (3)

c) El vídeo maestro ha finalizado su reproducción al alcanzar el final del vídeo y el vídeo esclavo continúa reproduciéndose hasta que este también finalice *(Figura 6-5)*.

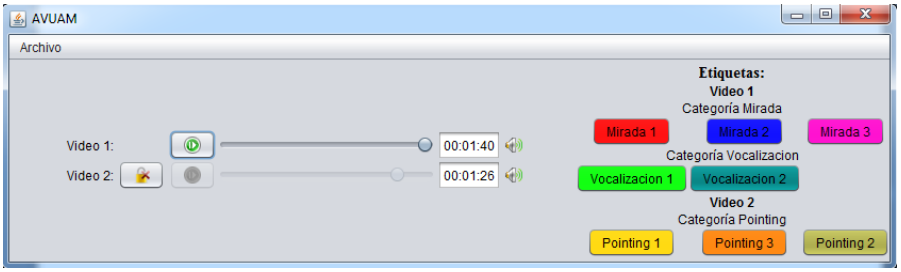


Figura 6-5: Vídeo maestro (Id 1) y vídeo esclavo (Id 2) sincronizados (3)

d) Tanto para el caso "a" como para el caso "c", si el usuario pincha en cualquier punto del slider del vídeo maestro la sincronización se mantendrá acorde a los tres casos posibles ya explicados, "a", "b" y "c".

Finalmente se ha realizado una serie de pruebas de rendimiento en las que se han reproducido 4, 6, 8, 10 y 12 vídeos simultáneamente en diferentes ejecuciones de la aplicación para comprobar que se mantiene una sincronización adecuada y que no se produzca ningún retraso entre los vídeos reproducidos. En estas pruebas los vídeos se han colocado en diferentes instantes de tiempo cada uno y se han sincronizado con el vídeo maestro. Posteriormente, se han realizado las siguientes pruebas:

- Cambiar el instante de tiempo del vídeo maestro para comprobar que el instante de los vídeos esclavos se mantiene sincronizado respecto al instante de tiempo marcado inicialmente.

- Durante la reproducción de los diferentes vídeos no se produzca ningún retraso o desfase de tiempo en ninguno de ellos.

- En el momento de aplicar etiquetas estas se apliquen correctamente y con los instantes de tiempo inicial y final correctos, así como los datos exportados se correspondan y sean correctos.

Tras la realización de estas pruebas los resultados han sido satisfactorios excepto para la prueba en la que se reproducen simultáneamente 12 vídeos. En esta prueba se ha podido comprobar que durante la reproducción simultánea de los 12 vídeos se aprecia un retraso o desfase entre ellos que aumenta a media que prosigue el tiempo de reproducción de los vídeos. Por ello, se recomienda como máximo la reproducción simultánea de hasta 10 vídeos.

Como punto final indicar que tras la finalización del proyecto se cumplen los objetivos planteados inicialmente. La herramienta ha sido probada y validada por miembros de LabLic, que la han recibido con mucho entusiasmo y que la van a utilizar en el futuro para sus investigaciones.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

Para este proyecto se han desarrollado dos módulos independientes: la aplicación principal que es una herramienta para la anotación de vídeos manual y un editor XML que permite al usuario editar los archivos de configuración XML que posteriormente serán cargados en la aplicación principal.

Es una aplicación que ayudará al psicólogo a estudiar las variables implicadas en el desarrollo del lenguaje y la comunicación infantil para niños comprendidos entre los 9 meses y 4 años de edad. Ha sido desarrollada para la facultad de psicología de la Universidad Autónoma de Madrid y adaptada a sus necesidades. Los requisitos de la aplicación han sido detallados en varias reuniones realizadas con ellos y han ido evolucionando a lo largo de estas reuniones.

Gracias a esta herramienta los psicólogos van a disponer de una aplicación sencilla de usar que les permita analizar y anotar los vídeos grabados durante las sesiones realizadas por los psicólogos con los niños. Una vez realizada la anotación de vídeos el psicólogo podrá exportar los datos a un fichero en diferentes formatos para posteriormente poder analizar los datos.

La aplicación cumple los objetivos establecidos en el inicio del proyecto además de ofrecer nuevas funcionalidades como el poder sincronizar en diferentes puntos de inicio cada vídeo y mejorando otras ya existentes como la sincronización de los vídeos a otras aplicaciones para la anotación de vídeos. Además, ha sido probada por los psicólogos de la facultad de psicología de la UAM y sus impresiones han sido buenas pareciéndoles una aplicación muy útil.

7.2 Trabajo futuro

En este proyecto se pueden realizar bastantes ampliaciones y/o mejoras, algunas de ellas pueden ser las siguientes:

- Implementar un analizador de los datos extraídos por la propia aplicación principal que permita sacar conclusiones o estadísticas. Este nuevo módulo ofrecería una gran ayuda y ahorro de trabajo al psicólogo.

- Desarrollar nuevas funcionalidades a la aplicación como una onda de audio, poder avanzar o retroceder frame a frame, ampliar zonas de un vídeo, poder configurar un tramo temporal de vídeo que se repita n veces configurable por el usuario, etc.

- Ampliar la complejidad de las etiquetas, añadiendo más información en el archivo XML que permita almacenar y manejar más información por cada etiqueta.

    - Añadir nuevos formatos de exportación como CHAT, CHILDES o formatos que puedan ser abiertos por otras aplicaciones de anotación de vídeos como ELAN.

Referencias

[1] LabLic, Laboratorio Infantil de Lenguaje y Comunicación, https://sites.google.com/site/lablic2014/home

[2] María del Mar Díaz, El Lenguaje Oral en el Desarrollo Infantil, http://www.csi-csif.es/andalucia/modules/mod_ense/revista/pdf/Numero_14/MARIA%20DEL%20MAR_DIAZ_2.pdf

[3] Eva Murillo, Nieves Galera y Marta Casla, Gesture and speech combinations beyond two-word stage in an experimental task, http://www.tandfonline.com/eprint/aHENCtqtkyYB5vSpbwc8/full

[4] S. Dasiopoulou, E. Giannakidou, G. Litos, P. Malasioti, and I. Kompatsiaris , PDF document of video annotation tools, http://mklab.iti.gr/files/AnnotationToolsOVerview.pdf

[5] Instituto Max Planck Institute for Psycholinguistics, The Language Archive, Nijmegen, The Netherlands, ELAN, URL http://tla.mpi.nl/tools/tla-tools/elan/

[6] Sloetjes, H., & Wittenburg, P. (2008).
Annotation by category – ELAN and ISO DCR.
In: Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008).

[7] Wittenburg, P., Brugman, H., Russel, A., Klassmann, A., Sloetjes, H. (2006).
ELAN: a Professional Framework for Multimodality Research.
In: Proceedings of LREC 2006, Fifth International Conference on Language Resources and Evaluation.

[8] Brugman, H., Russel, A. (2004).
Annotating Multimedia/ Multi-modal resources with ELAN.
In: Proceedings of LREC 2004, Fourth International Conference on Language Resources and Evaluation.

[9] Crasborn, O., Sloetjes, H. (2008).
Enhanced ELAN functionality for sign language corpora.
In: Proceedings of LREC 2008, Sixth International Conference on Language Resources and Evaluation.

[10] Lausberg, H., & Sloetjes, H. (2009).
Coding gestural behavior with the NEUROGES-ELAN system.
Behavior Research Methods, Instruments, & Computers, 41(3), 841-849.
doi:10.3758/BRM.41.3.591.

[11] Michael Kipp, Multimedia Annotation, http://embots.dfki.de/doc/Kipp%20MMIE%20preprint.pdf

[12] Michael Kipp, ANVIL,http://www.anvil-software.org/

[13] Kipp, M. (2014) ANVIL: A Universal Video Research Tool. In: J. Durand, U. Gut, G. Kristofferson (Eds.) Handbook of Corpus Phonology, Oxford University Press, Chapter 21, pp. 420-436

[14] Sun Microsystems, JMF API guide, http://docs.oracle.com/cd/E17802_01/j2se/javase/technologies/desktop/media/jmf/2.1.1/apidocs/

[15] Sun Microsystems, Inc, Java™ Media Framework API Guide, http://wwwinfo.deis.unical.it/fortino/teaching/gdmi0708/materiale/jmf2_0-guide.pdf

[16] Old Dominion University, JMF, http://www.cs.odu.edu/~cs778/jmflects/

[17] Jorge Lopez Aragoneses, Aplicaciones sobre SCTP (Stream Control Transmission Protocol). Streaming de video mediante JMF (Java Media Framework), http://jorgelopezaragoneses.blogspot.com.es/2010/02/sctp-stream-control-transmission.html

[18] Edgar Iván Durán Morales,Modulo de conexión para el CU Communicator vía WEB, http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/duran_m_ei/capitulo4.pdf

[19] Fabrice Bellard, FFMPEG, https://ffmpeg.org/

[20] SBGERMAN, Volatile, o como evitar usar bloqueos en concurrencia, http://www.art4software.com/2013/11/volatile/

Glosario

API	Application Programming Interface
JMF	Java Media Framework
Pixel	Es la menor unidad que forma parte de una imagen digital
Frame	Es una imagen individual en las que se divide un vídeo
Frame rate	Número de frames que se reproducen por segundo
Sincronizar vídeos	Coordinar dos o más vídeos para que presenten los datos multimedia de forma conjunta de tal forma que un vídeo tome el control de los demás vídeos
Vídeo maestro	Vídeo que tiene el control sobre lo/s vídeo/s esclavos
Vídeo esclavo	Vídeo cuyo control es gestionado por el vídeo maestro
FFMPEG	Librería que permite grabar y convertir vídeo, así como hacer streaming de audio y vídeo
Anotación vídeos	Añadir descripciones sobre un tramo de tiempo del vídeo
Etiqueta	Descripción que se puede añadir a un tramo de tiempo de un vídeo
URL	Universal Resource Locator
CSV	Comma-Separated Values
XML	eXtensible Markup Language
MOV	Formato de vídeo QuickTime
MPEG	Moving Picture Experts Group
AVI	Audio Video Interleave
MIDI	Musical Instrument Digital Interface
WAV	Waveform Audio File Format
JRE	Java Runtime Environment
JDK	Java Development Kit
Códec	Programa o dispositivo hardware capaz de codificar o decodificar una señal o flujo de datos digitales.

Anexos

A. Librería FFMPEG

FFMPEG es una librería de software libre que permite grabar, convertir y hacer streaming de audio y vídeo. En este proyecto se ha utilizado para convertir los vídeos facilitados por los psicólogos de la UAM a formato .mov. Este software puede ser descargado desde la página *https://ffmpeg.org/download.html* (Véase referencia [19]).

Esta librería soporta múltiples formatos de audio y vídeo de entrada como MPEG-1, MPEG-2, MPEG-4, H.261, H.263, H.264, x264, WMV, etc. En nuestro caso se ha utilizado para la conversión la biblioteca de vídeo x264, ya que es la que más óptima resultó para la reproducción de vídeos con la API JMF en comparación con otras bibliotecas como MJEPG.

Se ha necesitado utilizar esta librería para la conversión de los vídeos debido a que los vídeos facilitados provenían en formato .mp4 que no es compatible con la API JMF. Gracias a esta librería conseguimos mantener la calidad de vídeo y audio del vídeo original además de conseguir que sea totalmente compatible con nuestra aplicación.

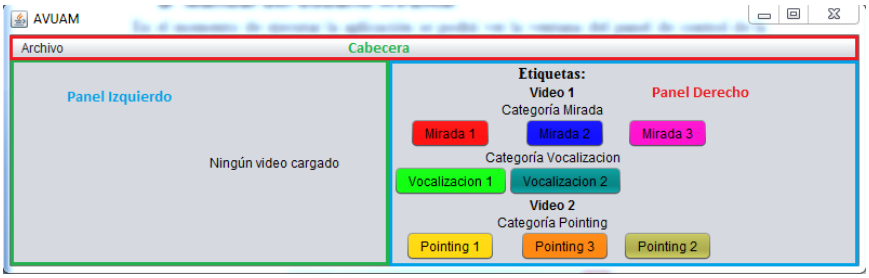
Para convertir los vídeos se ha utilizado la siguiente anotación:

> *ffmpeg -i "ruta vídeo entrada" -c:v libx264 -crf 23 -vcodec libx264 -acodec pcm_s16be -ac 2 -vf scale=800:500 -qscale 0 -y "ruta vídeo salida"*

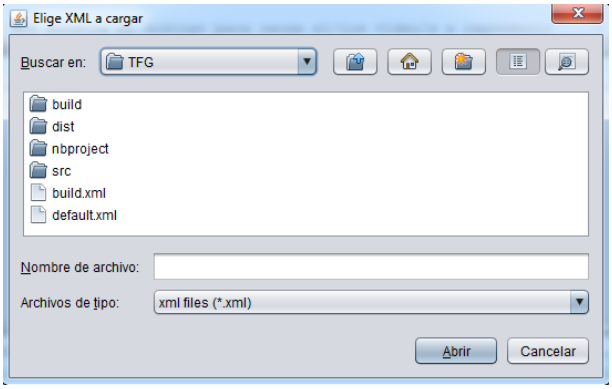
- **-i:** Nombre del fichero de entrada.
- **-c:v:** Librería con la que codificar el stream de vídeo de entrada. En este caso la librería x264.
- **-crf:** Es el factor de velocidad constante. Indica el número de bits que se van a utilizar por cada frame. El valor deberá estar comprendido entre 0 y 51 siendo el 0 el valor para la mejor calidad y el 51 el peor. En nuestro caso hemos utilizado el valor por defecto, 23.
- **-vcodec:** Establece el códec del vídeo. En nuestro caso x264.
- **-acodec:** Establece el códec de audio.
- **-ac:** Establece el número de canales de audio.
- **scale:** Escala la imagen al tamaño especificado en pixeles.
- **-qscale:** Sirve para establecer el nivel de la calidad del vídeo de salida. En este caso se pone el máximo nivel de calidad de audio y vídeo para el vídeo de salida. A más calidad mayor tamaño de fichero.
- **-y:** Parámetro para indicar que si ya existe el fichero los sobrescriba sin preguntar al usuario.

B. Manual del usuario AVUAM

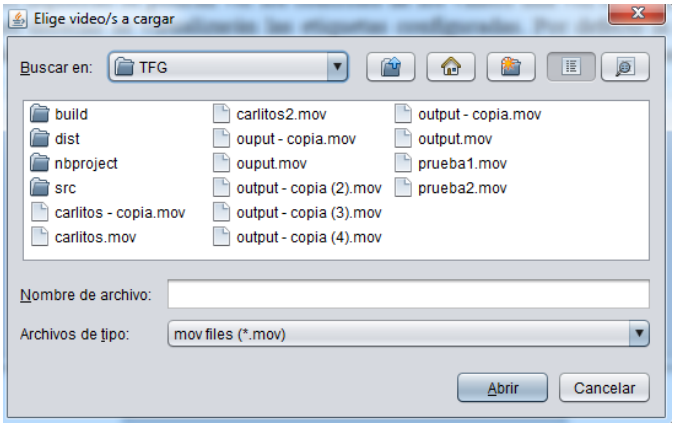
En el momento de ejecutar la aplicación se podrá ver la ventana del panel de control de la aplicación:



En el panel izquierdo se podrán ver los controles de los vídeos una vez se carguen estos y en el panel derecho se visualizarán las etiquetas configuradas. Por defecto la aplicación carga el fichero de configuración "default.xml". Para cargar otro archivo de configuración con otras etiquetas configuradas en este se deberá pulsar en "Archivo -> Cargar XML". En esta ventana seleccionaremos el fichero de configuración XML a cargar:



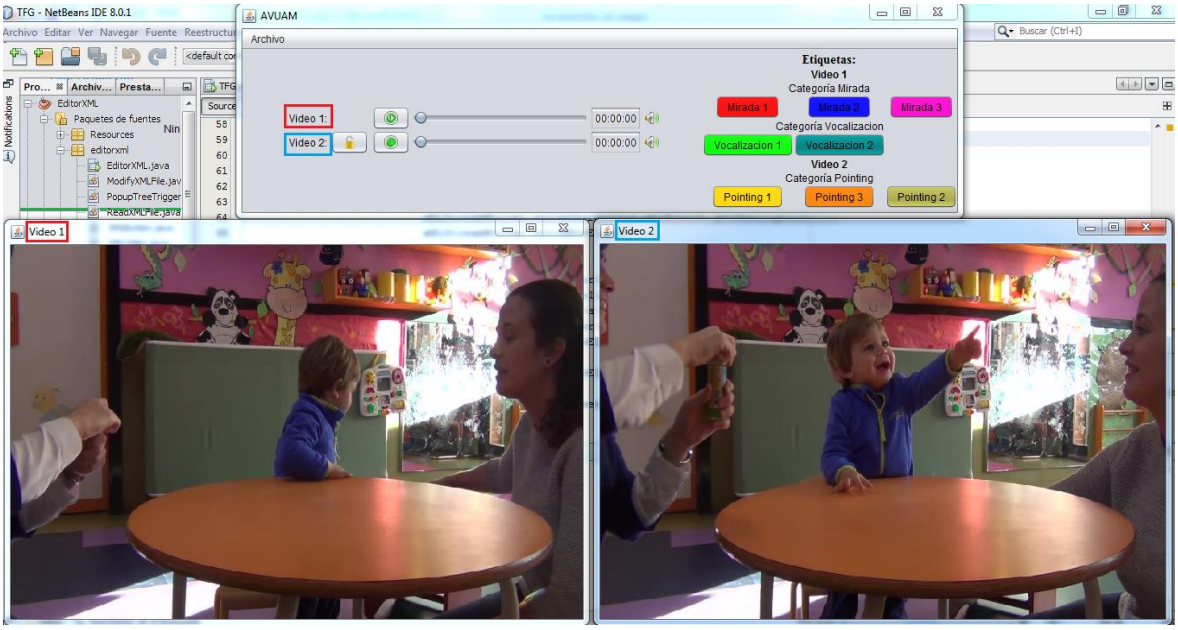
Una vez que tengamos cargado el archivo de configuración que deseamos se procederá a cargar los vídeos que se van a analizar. Para ello deberemos pulsar en "Archivo -> Cargar video/s":



En esta ventana elegiremos los archivos de vídeo a cargar. Si se quieren seleccionar varios se deberá pulsar la tecla "Control" + click izquierdo en el ratón en cada fichero de vídeo

que queramos seleccionar. Una vez se haya terminado la selección se deberá pulsar en "Abrir".

Tras cargarse los vídeos en la aplicación se visualizará la siguiente pantalla (Para este ejemplo se han seleccionado dos archivos de vídeo):

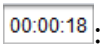


En la ventana principal de la aplicación aparecerán los controles asociados a cada vídeo donde el vídeo 1 corresponderá a la ventana con título "Vídeo 1" y el vídeo 2 a la ventana con título "Vídeo 2". La funcionalidad de los controles es la siguiente:

-  : Este botón solo aparecerá cuando se carguen dos más vídeos y sirve para sincronizar el vídeo asociado a este botón al vídeo uno. Podremos sincronizar y desincronizar este vídeo con el vídeo 1 tantas veces como queramos. El vídeo 1 nunca se podrá sincronizar con otro vídeo ya que siempre será el encargado de controlar los demás vídeos una vez que estén sincronizados con él.

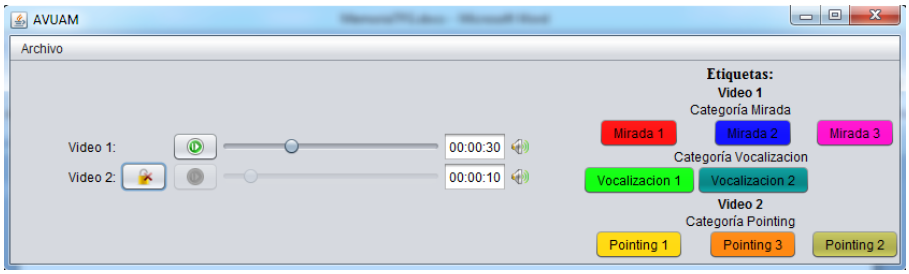
-  : Botón de reproducción/pausa. Cada vez que pulsemos en este botón el vídeo asociado se pausará o comenzará su reproducción dependiendo de si en ese momento se encuentra pausado o en reproducción.

-  : Indica en qué punto del vídeo se encuentra la reproducción del vídeo. Si pulsamos en cualquier punto de este control, el instante de reproducción cambiará al segundo en el que hayamos pulsado y se pausará el vídeo. También podremos arrastrar el "botón" del slider hasta el punto de reproducción deseado.

-  : Indica el segundo actual de reproducción del vídeo.

-  : Botón para activar/desactivar el audio del vídeo.

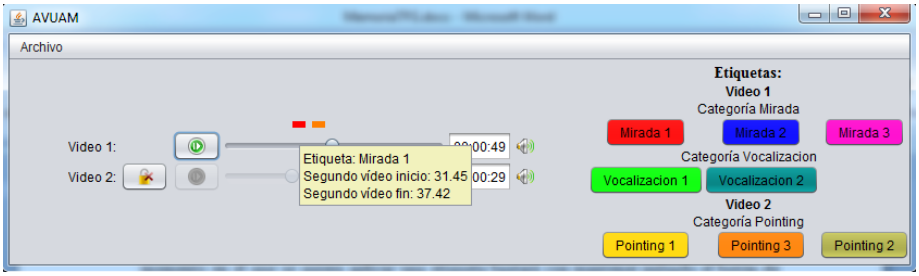
En el momento en el que el usuario desee sincronizar un vídeo deberá pulsar el botón anteriormente indicado. Tras pulsarlo los controles asociados a este vídeo se deshabilitarán, menos el control del audio que seguirá siendo gestionado individualmente por cada vídeo, y el control de este vídeo pasará a ser gestionado por el vídeo 1 tal y como podemos ver en la siguiente imagen:



En este momento si se cambia el instante de reproducción a través del slider o pausa/reproduce el vídeo 1 se verá replicado en los demás vídeos que se encuentren sincronizados con él. En el cambio del instante de reproducción siempre se mantendrá la sincronización entre el vídeo n y el vídeo 1 que se haya marcado al inicio de la sincronización de estos.

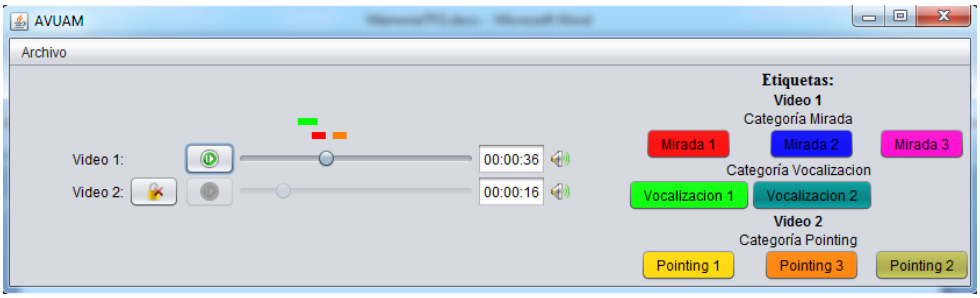
A Etiquetas

Las etiquetas van asociadas a un vídeo, y como consecuencia, en el panel de control de la aplicación veremos las etiquetas en el contenedor del vídeo al que se hayan asociado. En el momento en el que se quiera aplicar una etiqueta bastará con mantener pulsado el botón de la etiqueta deseada el tramo de tiempo en el que se quiera aplicar esta. Las etiquetas aplicadas serán pintadas en la parte superior del slider del vídeo 1 con el color que se haya configurado la etiqueta.

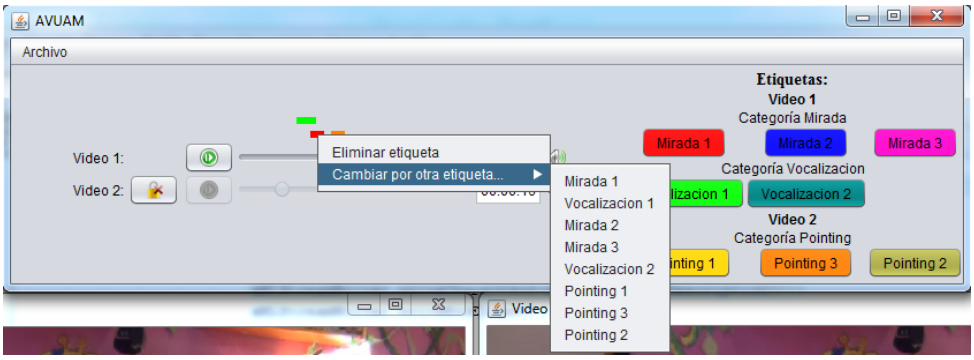


Si se mantiene el cursor del ratón encima de una etiqueta podremos ver un tooltip que indicará la descripción de la etiqueta aplicada, así como el instante de inicio y fin en el que se ha aplicado tal y como se puede ver en la imagen justo anterior.

Se pueden aplicar varias etiquetas en un mismo instante de tiempo. Si ocurre este caso las etiquetas que coincidan en un mismo instante de reproducción aparecerán apiladas una encima de otra siendo la superior la última etiqueta que se haya aplicado en ese instante de reproducción.

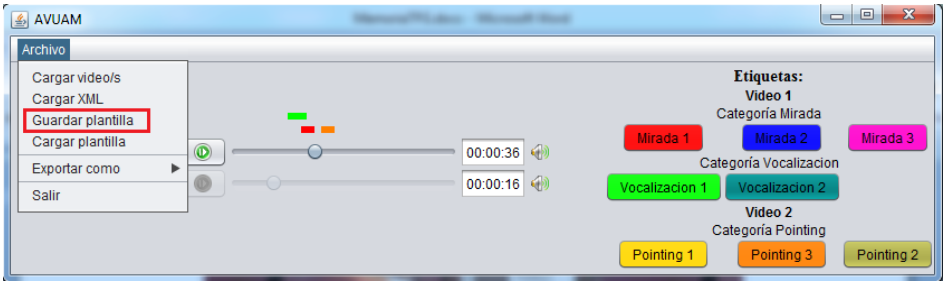


La aplicación añade una funcionalidad que permite eliminar o modificar una etiqueta ya aplicada en caso de que el usuario se haya equivocado al aplicar esta. Para ello se deberá pulsar click derecho sobre la etiqueta que se quiera modificar o eliminar. Al realizar esta acción aparecerá un menú en el que se podrá seleccionar la opción de eliminar etiqueta o de cambiar etiqueta, en cuyo caso se desplegarán todas las etiquetas configuradas en la aplicación para que el usuario pueda elegir la etiqueta por la que se sustituirá.



B Guardar y cargar plantillas

Si se desea posponer el análisis de los vídeos se podrá guardar el estado actual del análisis para posteriormente poder cargarlo en este mismo estado y proseguir con el análisis. Esta funcionalidad guardará los vídeos que se han cargado, segundo de reproducción en el que se encuentran, si están sincronizados o no y las etiquetas aplicadas. Para acceder a esta funcionalidad se deberá pulsar en "Archivo -> Guardar plantilla" en la que deberemos elegir el nombre y ruta donde se guardará la plantilla.



En cualquier momento se podrá cargar una plantilla desde la propia aplicación. Para ello se deberá pulsar en "Archivo -> Cargar plantilla" donde elegiremos la plantilla a cargar deseada.

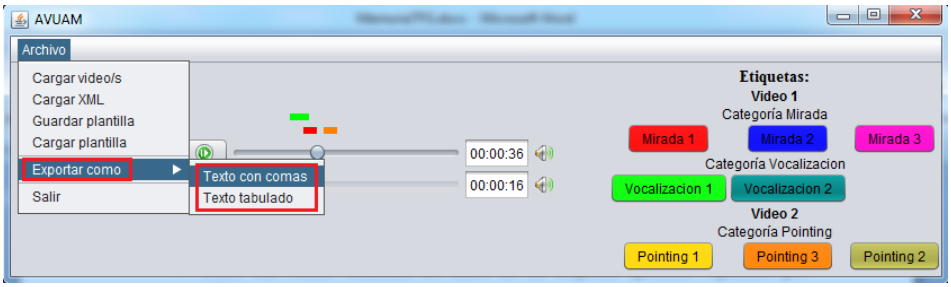
C Exportación de los datos

Una vez terminado el análisis de los vídeos podremos exportar los datos a un fichero en dos formatos diferentes:

- Texto con comas: Guardará los datos de las etiquetas aplicadas separados por comas.

- Texto tabulado: Guardará los datos de las etiquetas aplicadas separados con una tabulación.

Esta funcionalidad se encuentra en "Archivo -> Exportar como". Al llegar a este punto aparecerá un desplegable con los formatos anteriormente indicados en el que tendremos que elegir el formato con el que se exportarán los datos. Una vez elegido el formato deberemos indicar la ruta y nombre del fichero en el que se almacenarán los datos del análisis.

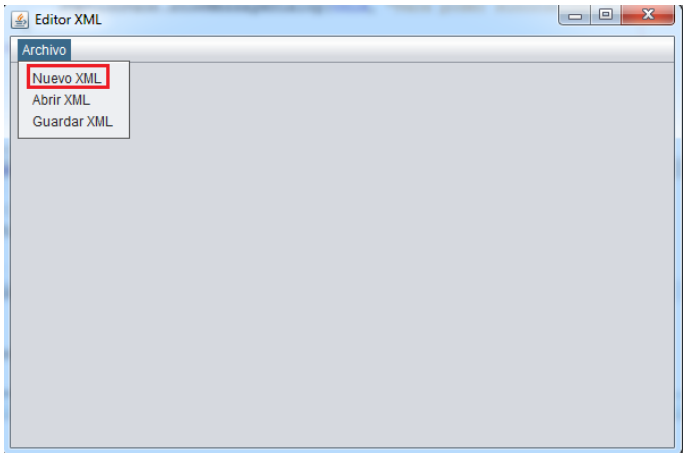


C. Manual del usuario Editor XML

En esta aplicación podremos modificar los archivos de configuración XML que posteriormente se podrán cargar en la aplicación principal de anotación de vídeos. En ella tendremos las funcionalidades necesarias para crear nuevos XML de configuración y la posibilidad de editar XML ya creados.

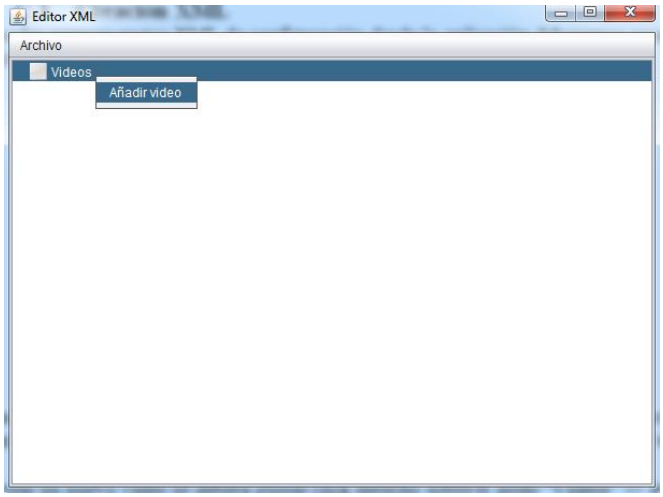
A Creación XML

Para poder crear un nuevo XML de configuración desde la aplicación deberemos acceder a "Archivo -> Nuevo XML".

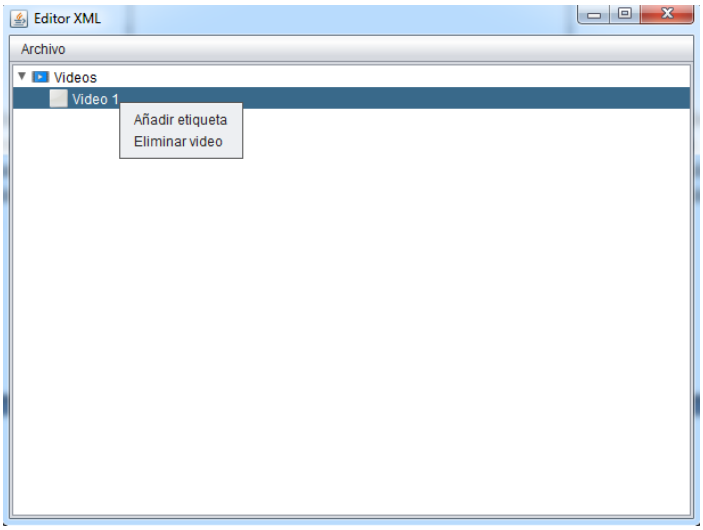


Al crear un nuevo XML en la aplicación veremos el nodo "Videos". A partir de este nodo podremos ir creando los vídeos y a partir de estos últimos las etiquetas asociadas al vídeo.

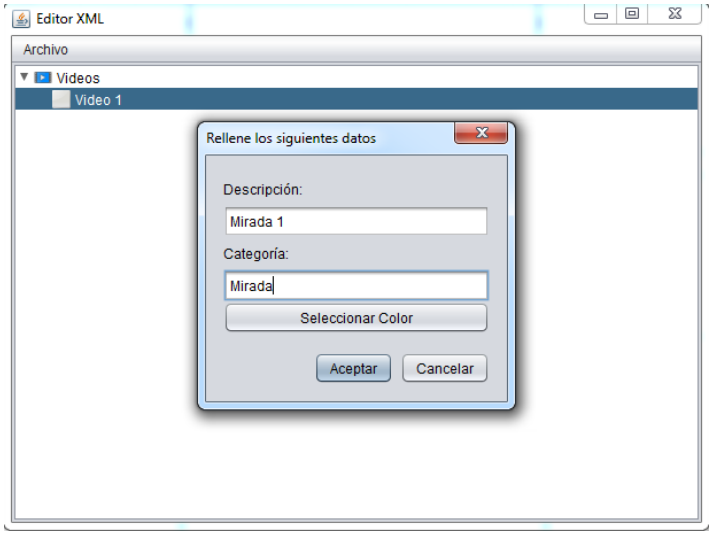
Para crear un nuevo vídeo se deberá pulsar click derecho sobre el nodo "Videos" -> "Añadir video":



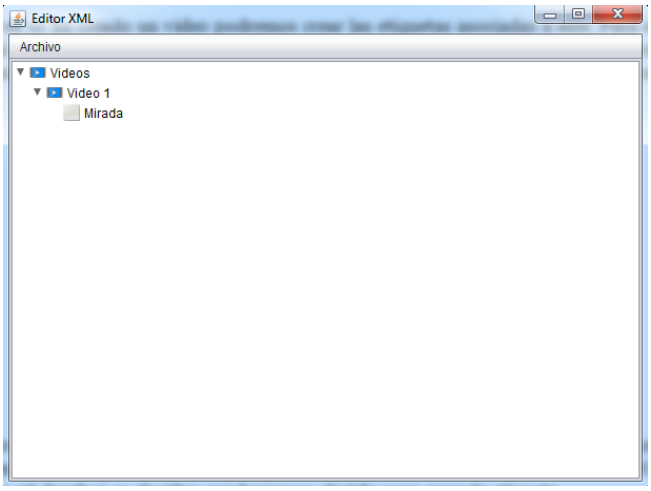
Al pulsar en "Añadir video" veremos que se ha creado un nuevo nodo llamado "Video 1". Podremos crear tantos vídeos como queramos, así como eliminar los ya existentes. Para eliminar un vídeo deberemos pulsar click derecho sobre el vídeo deseado a eliminar y "Eliminar video". Si se elimina un vídeo que contenga etiquetas también se eliminarán estas.



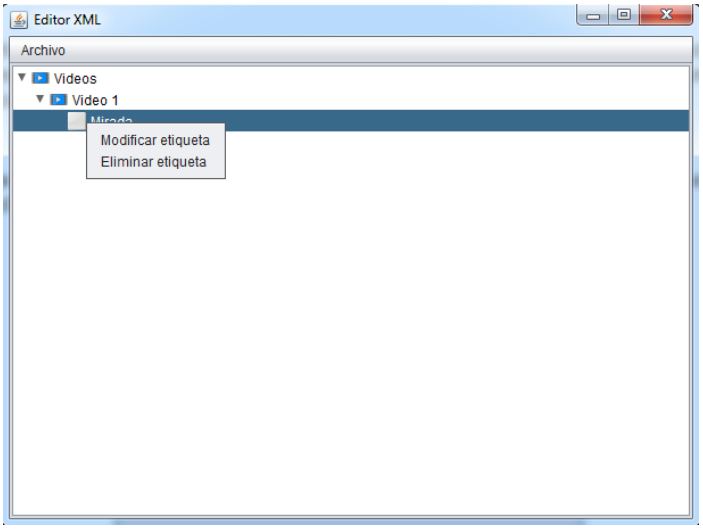
Una vez se ha creado un vídeo podremos crear las etiquetas asociadas a este. Para ello se deberá pulsar click derecho sobre el vídeo deseado -> "Añadir etiqueta". Aparecerá una ventana en la que tendremos que rellenar los datos de la etiqueta: descripción y color.



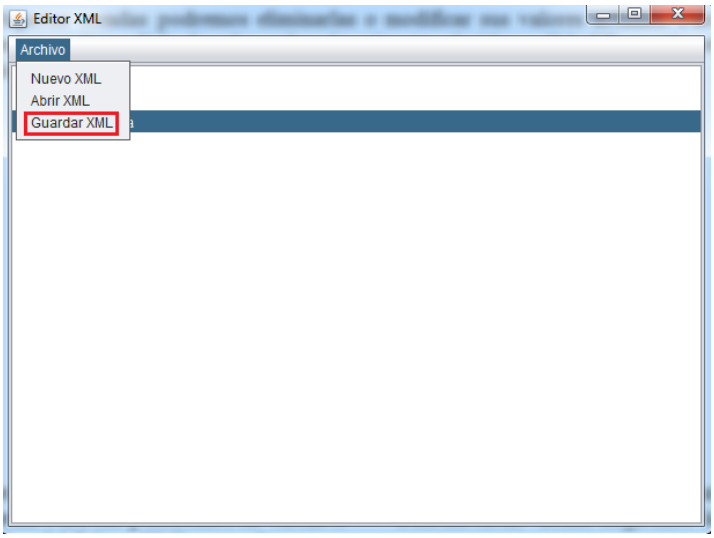
Pulsaremos "Aceptar" para crear la etiqueta o en caso contrario pulsaremos en "Cancelar". Si creamos la etiqueta veremos que aparece como un nuevo nodo descendiendo del nodo "Video n" donde n es el vídeo que hayamos elegido para crear la etiqueta.



Las etiquetas creadas podremos eliminarlas o modificar sus valores actuales. Para ello deberemos pulsar click derecho sobre la etiqueta elegida y "Modificar etiqueta" o "Eliminar etiqueta" según lo que se desee realizar.

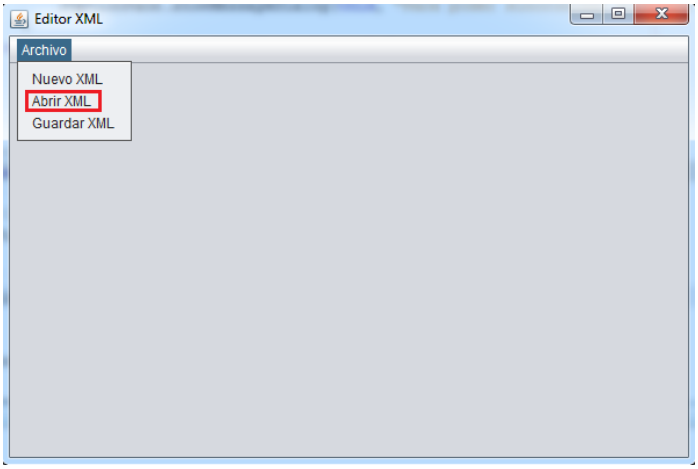


Una vez se ha terminado de editar el archivo de configuración será necesario guardarlo en el sistema. Para ello pulsaremos en "Archivo" -> "Guardar XML" donde elegiremos la ruta y nombre del fichero XML a guardar. En caso de existir ya un fichero con el mismo nombre nos dará la opción de sobrescribir el fichero ya existente o no.

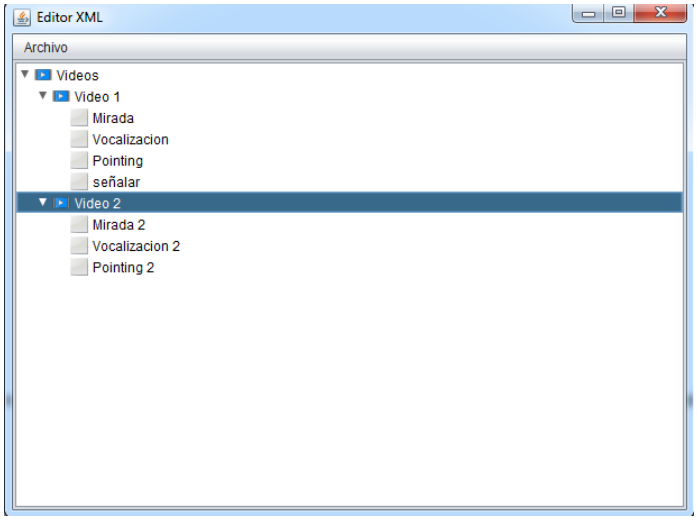


B Edición XML

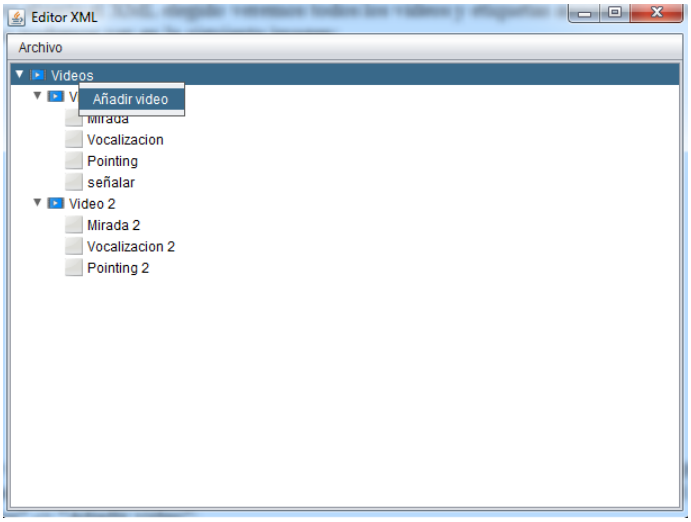
En caso de que se quiera editar un XML de configuración ya existente deberemos acceder a "Archivo -> Abrir XML".



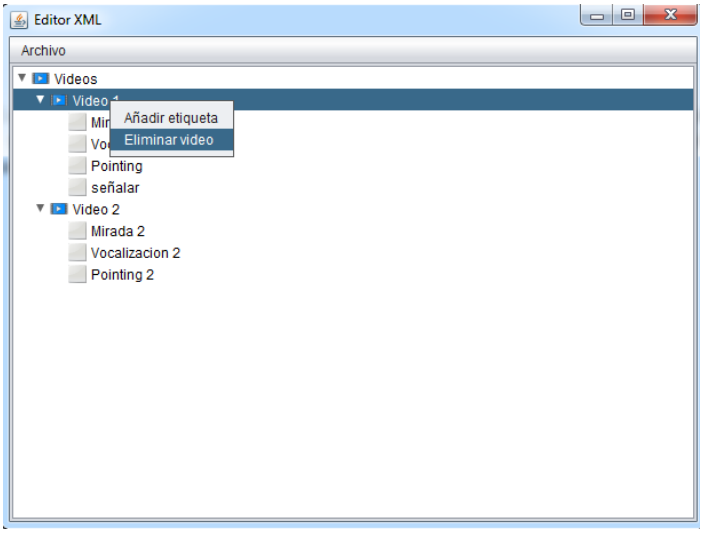
Una vez abierto el XML elegido veremos todos los vídeos y etiquetas asociados a estos tal y como podemos ver en la siguiente imagen:



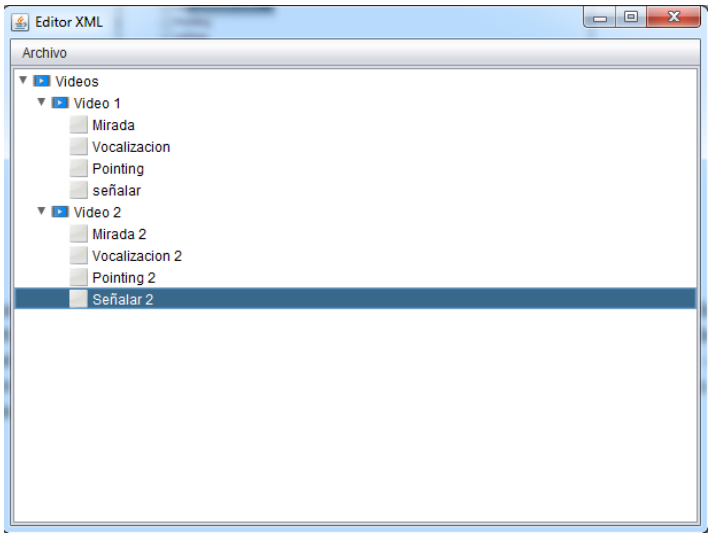
Desde esta ventana podremos eliminar o añadir etiquetas y vídeos y modificar etiquetas. En caso de que se quiera añadir un vídeo se deberá pulsar click derecho sobre el nodo "Videos" -> "Añadir video":



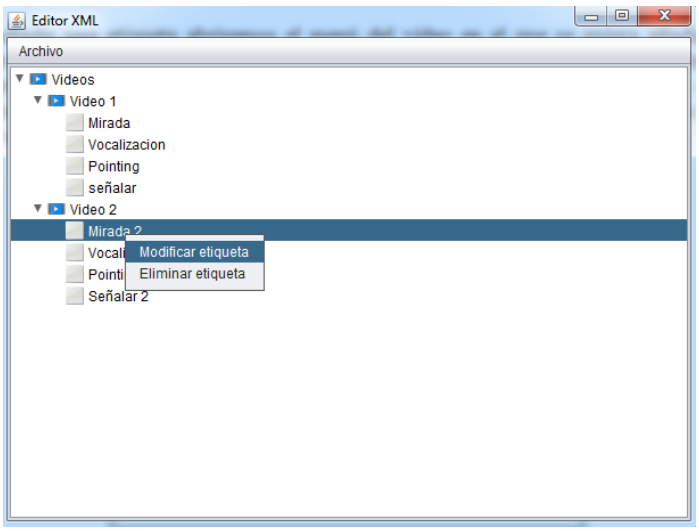
Por el contrario, si se desea eliminar un vídeo pulsaremos click derecho del ratón sobre el vídeo elegido a eliminar. Si se elimina un vídeo que contenga etiquetas también se eliminarán estas.



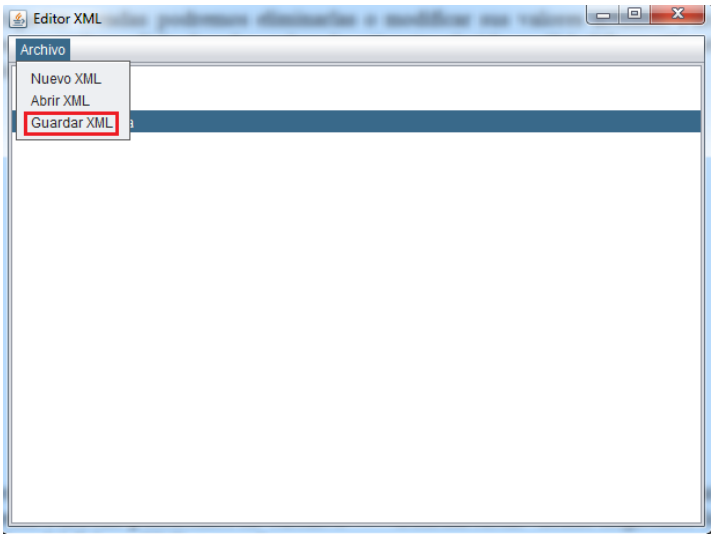
Para añadir una etiqueta abriremos el menú del vídeo en el que se quiera añadir una etiqueta y click sobre "Añadir etiqueta". Podremos ver una ventana en la que tendremos que seleccionar la descripción y el color de la etiqueta. Una vez rellenados los datos pulsaremos en "Aceptar" para crearla. La etiqueta creada la podremos ver como un nodo descendiente del vídeo sobre el que hayamos elegido crearla.



Si se desea eliminar o modificar una etiqueta ya existente se deberá pulsar click derecho sobre la etiqueta deseada y posteriormente pulsar en "Eliminar etiqueta" o "Modificar etiqueta" según la acción que se desee realizar.



Una vez se ha terminado de editar el archivo de configuración será necesario guardarlo en el sistema. Para ello pulsaremos en "Archivo" -> "Guardar XML" donde elegiremos la ruta y nombre del fichero XML a guardar. En caso de existir ya un fichero con el mismo nombre nos dará la opción de sobrescribir el fichero ya existente o no.


Editor XML
Archivo
Nuevo XML
Abrir XML
Guardar XML